CoLiAborative eMbedded networks for submarine surveillance (CLAM) Project

Deliverable D4.2

Underwater networking protocol design

Abstract

This deliverable addresses the problem of protocol design for the underwater acoustic networks to be employed in CLAM. The design concerns mainly Medium Access Control (MAC), routing and link-level error control, with considerations on data aggregation in hierarchical underwater networks. The analysis of the protocols considered in this deliverable is approached by means of simulations, using the nsMiracle+WOSS simulation package chosen in WP5. This deliverable contains several original contributions

- The simulation of three MAC protocols in the CLAM scenario; these protocols are based on three different types of random access in order to identify which approach works best;
- The design and simulation of a new protocol to enable the upload of data to a mobile sink patrolling a network of fixed nodes;
- The design and simulation of two different routing protocols;
- The derivation of an optimization framework which helps compute benchmarks for the average packet delivery delay in the network as a function of the number of nodes generating packets (under the assumption that the network topology and the mutual propagation delays are fully known);
- An analysis of the effects of data aggregation in hierarchically organized underwater networks in the presence of different transmission scheduling techniques.

In addition, we remark that some of the protocols designed for CLAM inherently operate according to cross-layer paradigms, either by requiring communication across different layers, or by incorporating the functions of more layers into a super-layer.

The analysis of the protocols above allows us to derive conclusions as to which protocol should be preferred as a function of the regime of operation required to the network. The results of this deliverable will be compared with those of D5.2, where we will simulate the CLAM protocols in a scenario of comparable size with respect
to that of the final demonstration. These considerations will help pave the way towards the final CLAM protocol stack (D4.3), which will be employed in the final demonstration.

The contributors to the present deliverable are:

- Paolo Casari, Michele Zorzi, Michele Rossi, Guido Cortelazzo (University of Padova (DEI))
- Chiara Petrioli, Daniele Spaccini (CINI)
- Giancarlo Bongiovanni, Francesco Lo Presti, Roberto Petroccia, Ariona Shashaj (University of Rome “La Sapienza” (URM))
- Salvador Climent, Nirvana Meratnia, Jose V. Capella, Paul Havinga (University of Twente (UT))

Kongsberg Maritime (KM) and SINTEF contributed by providing test area environmental parameters and the cNode data sheet.

DEI gratefully acknowledges the assistance and support of Federico Favaro and Giovanni Toso.

The final form of the document has been edited by Paolo Casari (DEI).

As this document has several contributors, it is inevitable that the writing style and level of detail varies throughout the report.

The delivery date of this document is: January 2012.

The dissemination level of this document is: Restricted to other program participants. This is motivated by the partially unpublished results that are contained in the deliverable. The deliverable can be made Public (in line with the CLAM DoW) as soon as the results are published, and no later than the end of the CLAM project.

Contents

Underwater networking protocol design ........................................... 1

1 Introduction ............................................................................. 5
2 MAC protocol descriptions ...................................................... 7
  2.1 Useful definitions .............................................................. 7
  2.2 CSMA ............................................................................. 8
  2.3 The Distance-Aware Collision Avoidance Protocol
      (DACP) ........................................................................ 9
  2.4 Tone-Lohi ................................................................. 11
  2.5 UW-Polling ................................................................. 13
2.5.1 Related work ........................................ 17
3 Routing protocol descriptions ........................................ 18
  3.1 The Source routing for Underwater Networks (SUN) protocol ........................................ 19
    3.1.1 Protocol Overview ........................................ 19
    3.1.2 The SUN algorithm ........................................ 20
    3.1.3 Path Establishment Algorithm ................................ 22
    3.1.4 Discussion on Routing Metrics ................................ 22
    3.1.5 SUN protocol details ........................................ 23
  3.2 SUN vs DSR ........................................ 26
  3.3 Channel Aware Routing Protocol (CARP): a cross-layer approach for underwater routing ........................................ 28
    3.3.1 Protocol overview ........................................ 28
    3.3.2 Protocol description ........................................ 28
4 Simulation scenarios and parameters ........................................ 31
5 Metrics of interest ........................................ 38
  5.1 Scenario 1 ........................................ 38
  5.2 Scenarios 2 and 3 ........................................ 38
6 Optimal packet size selection ........................................ 40
  6.1 System Model ........................................ 41
  6.2 Simulation scenarios and settings ........................................ 41
  6.3 Performance results ........................................ 43
  6.4 Varying the deployment scenarios ........................................ 44
7 MAC protocol simulations ........................................ 47
  7.1 CSMA without error control ........................................ 47
  7.2 CSMA with S&W ARQ ........................................ 52
  7.3 DACAP without error control ........................................ 57
  7.4 DACAP with S&W ARQ ........................................ 62
  7.5 Tone-Lohi without error control ........................................ 66
  7.6 Tone-Lohi with S&W ARQ ........................................ 70
  7.7 On the effect of S&W ARQ on larger scenario ........................................ 74
    7.7.1 CSMA with S&W ARQ ........................................ 76
    7.7.2 DACAP with S&W ARQ ........................................ 81
    7.7.3 Tone-Lohi with S&W ARQ ........................................ 86
  7.8 Concluding remarks on the simulations of CSMA, DACAP and Tone-Lohi ........................................ 91
  7.9 UW-Polling simulations ........................................ 91
    7.9.1 Introduction ........................................ 91
    7.9.2 A note on the simulations ........................................ 92
    7.9.3 Results ........................................ 95
    7.9.4 Uw-Polling with AUVs moving at 8 knots ........................................ 99
    7.9.5 Concluding remarks on UW-Polling simulations ........................................ 105
8 Routing protocol simulations ........................................ 105
  8.1 SUN ........................................ 105
8.2 CARP ........................................... 114
  8.2.1 Protocol parameters setting ...................... 114
  8.2.2 Results ....................................... 114
9  Optimal Periodic Joint Routing and Scheduling ............. 118
  9.1 Related work .................................... 121
  9.2 System Model .................................... 122
    9.2.1 Network Model ................................ 122
    9.2.2 Traffic Model ................................ 122
    9.2.3 Conflict and Interference Model ................. 123
  9.3 Optimization Model ................................ 124
    Variables ........................................ 125
    Objective Function ................................ 126
    Constraints ....................................... 126
    Sink constraints ((24)-(25)) ....................... 126
    Conflict Constraints ((26)-(29)) .................. 126
    Buffer management (30)-(32) ....................... 130
  9.4 Centralized heuristic ................................ 132
    9.4.1 Routing/Conflict graph ....................... 132
    9.4.2 Routing ....................................... 133
    9.4.3 Scheduling .................................... 135
    Ranking Metrics .................................... 138
  9.5 Performance Evaluation ............................. 139
10 On the impact of different scheduling and retransmission techniques on hierarchical aggregation-aware routing in underwater networks ........................................... 144
  10.1 The EDETA protocol .............................. 145
  10.2 Scheduling and Retransmission techniques ............ 145
  10.3 Performance evaluation ............................ 147
    10.3.1 TnoAck vs. DnoAck ......................... 148
    10.3.2 TAck vs. DAck ............................. 149
    10.3.3 DAck vs DnoAck2 ........................... 150
  10.4 Conclusions ..................................... 150
11 Discussion and Conclusions .......................... 151
References ................................................................ 152
1 Introduction

This deliverable addresses the problem of protocol design and evaluation for the underwater acoustic networks to be employed in CLAM. The design concerns mainly Medium Access Control (MAC), routing, transmission scheduling and link-level error control.

Three different MAC protocols are considered in order to compare several design concepts based on random access. Namely, we evaluated a version of ALOHA called CSMA which includes short carrier sensing phases to avoid trivial collision events; the DACAP protocol, which is representative of Carrier Sense Multiple Access–Collision Avoidance (CSMA/CA) protocols based on RTS/CTS exchanges; Tone-Lohi, which simplifies the channel access procedure and does not involve the exchange of packets requiring only busy tones to be sent.

In accordance with the oil well monitoring application considered in CLAM, the protocols have been designed to work on static networks. The only exception to this case is the presence of one mobile node acting as the sink, (e.g., an Autonomous Underwater Vehicle, AUV), being sent to patrol an area or to investigate possible incidents by directly inspecting sensor readings, in case the transport of data to a control center is malfunctioning. For this special case, we specifically designed UW-Polling, a protocol that allows the mobile to retrieve data from the static nodes using a polling procedure.

Since the networks of interest in CLAM are typically multihop, we designed two different protocols for multiuser, multihop networks, called respectively SUN and CARP. These protocols are designed according to different concepts and employ different metrics. SUN is a reactive source routing protocol that relies on CSMA at the MAC level (CSMA which will be shown in this deliverable to offer the best performance in a one-hop scenario). CARP selects relays hop-by-hop preferring links with a good history of successful transmissions (among other metrics for choosing the next hop) and features a relay selection procedure that is integrated with channel access.

Notably, in the protocols above there are inherent cross-layer interactions: for example, MAC protocols must be notified by the physical layer as to whether or not a reception is in process before accessing the channel. This carries over to SUN, that builds on CSMA. As another example, the CARP routing protocol is integrated with channel access, in a different form of cross-layer interaction (i.e., the integration of two layers into a super-layer).

A cross-layer Integer Linear Programming (ILP) framework for routing and transmission scheduling in underwater networks has been extended based on a contribution in the literature. The extension includes the use of heuristics to make the problem scale better with the network size. The framework can be employed, e.g., to derive an estimate of the average delivery delay in the network.

Finally, as a preliminary step for the evaluation of network performance in the presence of data aggregation, a set of simulation results is presented concerning the application of several transmission scheduling techniques to the EDETA-e routing protocol. EDETA-e organizes the network in a hierarchical manner, such that the
nodes are grouped in clusters and routing operations can be handled by the clusterheads, who become therefore the ideal location for data aggregation.

This document is organized as follows. In Sections 2 and 3 we introduce the design of MAC and routing protocols, respectively.

In Section 4 we describe the simulation scenarios and the parameters thereof. We also discuss the parameters of the transmission system, which are akin to those of the modem used in CLAM, i.e., Kongsberg Maritime’s cNode mini. These parameters and the simulation scenario have been chosen in collaboration with WP2, WP3 and WP5.

Section 5 introduces the metrics of interest for the evaluation of the protocol performance. These metrics have been chosen in collaboration with WP2, and contain such indications as the energy consumed by protocol operations. This is a substantial step towards the estimation of the maximum duration of those CLAM sea trials that explicitly require communications among battery-powered nodes.

Section 6 shows the impact of the packet size on the performance of underwater MAC protocols, and motivates our choice to perform several network protocol simulations for different packet sizes.

Section 7 reports the simulation results for the MAC protocols introduced in Section 2, whereas Section 8 presents the simulation results related to routing protocols.

Section 9 presents the ILP framework for optimizing routing and scheduling in underwater networks, and the improvements and heuristics introduced to make it scale well with the network size. A performance evaluation shows the good accuracy of the improved version.

Section 10 presents preliminary results on the effectiveness of transmission scheduling techniques in hierarchical underwater networks employing data aggregation.

Finally, we summarize the results of this deliverable and draw some concluding remarks in Section 11.

There are several original contributions in this deliverable. The original contributions, in terms of protocol design, are represented by

- UW-Polling (Section 2.5), a protocol for efficiently uploading data from a network of fixed sensors to a mobile AUV;
- SUN (Section 3.1), a reactive source routing protocol for underwater networks;
- CARP (Section 3.3), an integrated MAC and routing protocol that relies on a hop-by-hop relay selection procedure.

The other MAC protocols described in Section 2 are taken from the literature, but had to be implemented in order to be tested in CLAM scenarios. For these protocols, the test in the scenarios of interest in CLAM are original contributions.

Further original contributions are represented by

- The derivation and extension of an ILP framework with heuristics for fast and scalable derivation of near-optimal routing and scheduling in underwater networks;
- The test of several retransmission policies in conjunction with the EDETA-e protocol, for aggregation-aware, hierarchically organized underwater networks.
Before proceeding to the description of the protocols and the presentation of the results, we emphasize that this deliverable is not aimed at evaluating the performance of the protocols in a specific scenario to be employed in the final demonstration of CLAM. Here, we focus on the description of the protocols, on their performance evaluation, and on general considerations on their suitability for the purposes of CLAM. This is a necessary step towards the test of the protocols in a likely deployment for the final CLAM demonstration in D5.2, and towards the definition of the final protocol stack in D4.4.

2 MAC protocol descriptions

In this section, we briefly introduce and describe the MAC protocols we have considered for possible use in the CLAM project. We will focus on four different protocols:

- a version of CSMA featuring a short clear channel assessment of random duration [1];
- the “aggressive” version of the Tone-Lohi protocol [2];
- the Distance-Aware Collision Avoidance Protocol (DACAP) [3].
- UW-Polling

The first three protocols have been chosen from the literature due to the different level of handshaking the require for the coordination of channel access procedures. In particular, CSMA presents no coordination, except for a short channel sensing phase to avoid starting a transmission while another acoustic signal is already propagating towards the receiver; Tone-Lohi is a transmitter-driven contention-based protocol which does not require heavy signaling in the presence of low traffic, but is likely subject to the hidden terminal problem, because receivers take no part in preliminary signaling; finally, DACAP is a full-fledged 3-way handshaking protocol (4-way in case acknowledgments are employed to detect transmission errors), with further refinements to avoid some collision events.

UW-Polling, instead, is a new protocol designed for CLAM. It allows a mobile Autonomous Underwater Vehicle (AUV) to retrieve data packets from a network of fixed sensors using a polling mechanism. The protocol descriptions, along with relevant protocol parameters, are provided in the following subsections.

2.1 Useful definitions

For clarity, we report hereafter a set of useful definitions that are going to be referred to in the rest of the deliverable whenever there is a possible ambiguity.

- **Packet**: the collection of the data generated by the application layer and the underlying protocol stack headers.
- **Payload:** the data generated by the application layer
- **MAC frame** (or simply **frame**): any MAC-level control packet (such as RTS, CTS, ACK, etc.)
- **MAC layer fragment:** in case a Frame need to be fragmented in accordance to the MAC-level Maximum Transmission Unit (MTU), each section of fragmented frame is a MAC-level fragment

### 2.2 CSMA

**CSMA** By CSMA we mean a form of 1-persistent CSMA protocol. It prescribes that nodes sense the channel for a short, random-length time, in order to avoid a trivial collision event: assume that nodes A, B and C are roughly aligned, and A transmits to C. If B has a transmission for any other node, and starts the transmission while the signal from A is propagating over B, the two signals would very likely collide at C. The short channel sensing provides a means to avoid this situation. In case the channel is sensed busy, the node keeps listening until the transmission is completed, as this does not consume too much power [4]. Immediately thereafter, a further channel sensing phase (again of random length and short) is scheduled, and the process goes on until the channel is finally sensed clear, after which the transmission is issued. In the literature this protocol was called CSMA-ALOHA [5] because it carries the advantage of both protocols, such as the simplicity, the absence of overhead and the capability to sense harmful interference in some cases.

CSMA comes in two flavors, with and without acknowledgement (ACK) packets for error control via a Stop-and-Wait Automatic Repeat reQuest (S&W ARQ) policy. In the S&W ARQ version, the node wait the ACK of every packet transmitted, setting a timeout. if this timeout expires without receiving the ACK packet, he sends back the packet one more time. The ACK timeout can be selected by the user, and than it is updated by the protocol based on the values of the round-trip time (RTT) between the transmitter and the receiver (in order to take in account eventually change of SSP or channel condition). CSMA is feasible to be used with complex routing protocol in a multi-hop with a lot of control signaling (like SUN) because offers high throughput and low latency with reasonable PER especially in case of low traffic.

A selection of the most important protocol parameters follows:

- **ACK\_timeout:** 5 s. This is a multiplier value useful for the calculation of the back-off time with the formula below
- **backoff\_tuner:** 1 s. This is another multiplier value useful to calculate the back-off time.
- **listen\_time:** 0.5 s. This is the duration of the sensing phase.
- **ACK\_size:** 10 byte.

The back-off time is calculated as follows:
\[ \text{backoff} = \text{backoff\_tuner} \cdot r_v \cdot 2 \cdot \text{max\_prop\_delay} \cdot 2^{\text{backoff\_counter}} \]

where \( r_v \) is a value chosen uniformly at random in the interval \([0, 1]\), \( \text{backoff\_tuner} \) is an integer multiplier value chosen by the user and \( \text{backoff\_counter} \) is a value that is increased every time a back-off time is calculated.

### 2.3 The Distance-Aware Collision Avoidance Protocol (DACAP)

The Distance-Aware Collision Avoidance Protocol (DACAP) \([3]\) employs a Request-To-Send/Clear-To-Send (RTS/CTS) handshake mechanism to reserve the channel for packet transmission, extending this common procedure with a method to accommodate for the longer delays of underwater links. More specifically, when a node has a data packet to send, it performs channel sensing, and it transmits an RTS if the channel is idle. If the channel is instead sensed busy, the sender computes a back off time and after this time checks the channel again. Upon correctly receiving an RTS packet, the destination node, if idle, replies with a CTS. It then waits for the data packet, which can be acknowledged or not, depending on the chosen version of the protocol \([3]\). DACAP adapts to the underwater sound propagation delays by using a warning mechanism, as follows. If, while waiting for a data packet, a destination node overhears a control packet intended for some other node, it sends a very short WARNING packet to its wanted sender. This WARNING serves as an alert about possible interference that could affect the upcoming communication. Upon receiving a CTS packet, a sender waits for a time \( T_{\text{warning}} \) before transmitting the data packet. This time \( T_{\text{warning}} \) is defined as the minimum time to detect interference from neighboring nodes within a certain range. The computation of \( T_{\text{warning}} \) depends on the propagation time between the source and the destination (estimated by the sender by measuring the RTS/CTS handshake time) and also on other factors concerning the distance of potential interferers.

A second mechanism to avoid collisions prescribes that if, while waiting for a CTS, the sender overhears a control packet, it aborts the data communication. The communication can also be aborted if, during the warning period \( T_{\text{warning}} \), the sender receives a WARNING packet from the destination, or overhears a control packet from other nodes. In these cases the sender computes a back-off time and restarts the channel access procedure from scratch at a later time. A maximum number of attempts is defined, after which the packet is dropped.

We note that a receiver, after sending a WARNING packet, does not know if this packet has reached the sender on time to make it abort the data transmission. Hence, the receiver keeps listening to the channel because the data packet could still be received correctly.

\( T_{\text{warning}} \) is defined as
The minimum hand-shake length is represented by $t_{\text{min}}$, which is used to optimize the protocol operations for a given network. For a network where most links are close to the transmission range, $t_{\text{min}}$ should be nearly as large as the RTT corresponding to the maximum range $T$. When some links are shorter, this time can be reduced. The distance between two nodes performing a hand-shake is indicated by $U$, and $U + \Delta D$ is the minimum distance to an interfering node for which correct reception is still possible (because the signal-to-noise-and-interference ratio is still sufficiently high). The corresponding propagation times are obtained by dividing the distances by the speed of sound underwater, denoted here by $c$. The time $t_{\text{data}}$ denotes the duration of a data packet. There is an additional restriction that $T_{\text{warning}} > 2\Delta D/c$ to avoid collisions with control packets.

In case DACAP is supplied with a S&W ARQ mechanism requiring ACK packets to confirm correct data reception, the sender backs off and retries if no ACK is received within a given time after the transmission of a data packet. The same happens if, while waiting for the ACK, the sender instead overhears an RTS, a CTS or a DATA packet from other nodes. Potential interferers are blocked as usual in RTS/CTS schemes. When ACKs are employed, the $T_{\text{warning}}$ time has to be modified to take into account the transmission of ACK packets

$$T_{\text{warning}} = \begin{cases} t_{\text{min}} - 2U/c, & U/c < t_1 \\ 2(U + \Delta D)/c - t_{\text{min}}, & U/c \in (t_1, t_2) \\ 2\Delta D/c - t_{\text{data}}, & U/c > t_2 \end{cases}$$

(2)

where

$$t_1 = \frac{t_{\text{min}} - \min(\Delta D/c, t_{\text{data}})}{2}$$

$$t_2 = \frac{t_{\text{data}} + t_{\text{min}}}{2}$$

(3)

Three new parameters have to be introduced:

- $T$ is the expected maximal transmission range.
• $T_{\text{min}}$ represents the minimum time that any transmitter will wait from when the CTS is received to when the data packet is sent. It must be longer than $2\Delta D/c$ and shorter than both $T$ and $t_{\text{min}}$.

• $\Delta t_{\text{data}}$ is the maximum difference between the duration of two data packets. If there is no minimum data packet size, then $\Delta t_{\text{data}}$ equals the maximum data packet duration. $\Delta t_{\text{data}} = 0$ if the packet size is constant.

We anticipate here some parameter values that will be used in the simulations and refer the reader to Section 4 for a description of the simulation scenarios, and in particular of scenario 1. The parameters chosen for the simulations in this deliverable are as follows:

• $\maxpropdelay : 0.6579 \text{ s}$. This value is calculated assuming that the maximum distance from the sink to a node is on the order of 1 km and the propagation speed is 1500 m/s. This value is useful to calculate the following $t_{\text{min}}$ value and also to calculate the back-off timer of a node

• $t_{\text{min}} : 1.07 \text{ s}$. This value is defined as the minimum time needed to make an hand-shake among two nodes. It is calculated assuming that most of the nodes are nearest to the sink than 1 km. In this manner we can take less time to do an hand-shake procedure and increase the throughput. So, this value is little less than twice the maxpropdelay.

• $\Delta D : 250 \text{ m}$. In this scenario the network is not so large and a control message sent from a node can reach easily all the nodes in the network. Hence, a small value can be chosen in order to decrease the protocol response time.

• $\text{backoff tuner} : 1$. This is a multiplier value that is useful for the calculation of the back-off as written in the formula below

• $\text{CTS size} : 48 \text{ byte}$

• $\text{RTS size} : 48 \text{ byte}$

• $\text{ACK size} : 48 \text{ byte}$

• $\text{WARNING size} : 48 \text{ byte}$

The backoff time is a uniform random variable and is calculated as follows:

$$\text{backoff} = \text{backoff tuner} \cdot r_v \cdot 2 \cdot \maxpropdelay \cdot 2^{\text{backoff counter}} \quad (4)$$

where $r_v$ is a value chosen uniformly at random in $[0, 1]$, $\text{backoff tuner}$ is a integer multiplier value chosen by the user and $\text{backoff counter}$ is a value that is increased every time a back-off time is calculated, up to a maximum of 5 times.

### 2.4 Tone-Lohi

Tone-Lohi [2] is a protocol for single-hop underwater networks that employs a transmitter-driven negotiation based on the transmission of tones in order to reserve the channel. The protocol works as follows: When a node has a data packet to transmit, it first starts a reservation period (RP). An RP is composed of a given number
of slots called contention rounds (CRs). During a CR, the sender transmits a short tone\(^1\) to inform other nodes about its intention to access the channel. After that, it listens to the channel to detect if other nodes also have data packets to send. The presence of possible contenders for channel access is detected by listening to the tones they would send during the same CR. Each node contending for the channel counts how many other tones it receives during a CR, and hence infers how many nodes are currently contending for channel access. If no other tone is heard during the CR, the RP is over, the node seizes the channel and starts transmitting the data packet. If contention occurs, the contenders back off for a number of CRs chosen randomly and uniformly between \([0, N]\), where \(N\) is the number of competitors. A RP continues until successful channel access. We stress that the nodes need not be synchronized. Each node that has data packets to send starts a RP for channel access and transmission, independent of other nodes. The duration of a CR is set so that a node has sufficient time to detect as many contenders as possible. Different flavors of the Tone-Lohi protocol are described in [2].

- **Synchronized Tone-Lohi (ST-Lohi):** All nodes are time synchronized. Synchronizing each contention round simplifies the operations of the protocol, at the cost of requiring the distribution of some reference time among the nodes.
- **Conservative Unsynchronized Tone-Lohi (cUT-Lohi):** The nodes are not synchronized, and the CR duration accounts for worst-case tone arrival times, assuming that tones may be transmitted from within twice the maximum range from which a tone would still be heard. Therefore, each CR lasts twice the maximum propagation delay plus twice the transmission time of a tone.
- **Aggressive Unsynchronized Tone-Lohi (aUT-Lohi):** Nodes are not synchronized and the CR duration accounts only for one way communications. Therefore, each CR lasts for the maximum propagation delay plus the transmission time of a tone.

Among these flavors, aggressive Tone-Lohi is the one that maximizes the channel utilization. In the aggressive version of Tone-Lohi the CR lasts for the time needed to transmit a tone packet plus the maximum anticipated propagation delay. Figure 1 shows the organization of the Tone-Lohi protocol frame. Since several indications exist that the aggressive Tone-Lohi version performs better than the conservative one (e.g., see [1]), we will consider aggressive Tone-Lohi for the comparative evaluation.

---

\(^1\) Tones may be implemented in multiple ways. The method devised in [2] assumes the presence of dedicated hardware in the nodes. When this is impossible (as in the CLAM nodes), the tones can be implemented as very short packets, transmitted on the same band as the data packets. This is the way we implemented tones in this deliverable.
of MAC protocols in what follows. As for DACAP and CSMA, Tone-Lohi comes in two versions: one without error control, and one with S&W ARQ.

We anticipate here some parameter values that will be used in the simulations and refer the reader to Section 4 for a description of the simulation scenarios, and in particular of scenario 1. For the simulation in this scenario _aUT-Lohi_ has been used. The most important parameters follow:

- _max prop delay_: 0.6579 s. This value is calculated assuming that the maximum distance from the sink to a node is on the order of 1 km and the propagation speed is 1500 m/s.
- _ACK size_: 24 byte
- _ACK timeout_: 2 s. If the sender does not receive an ACK within this time, it re-sends the packet (according to a S&W scheme).

The back-off time is calculated as

\[ \text{backoff} = r_v \cdot CR_{duration} \cdot \max(1, \text{curr\_contenders}) \]  \hspace{1cm} (5)

where _CR_{duration} is the duration of the Contention Round and is calculated as \(2 \cdot \text{max prop delay} + TONE_{PKT\_time} \). _TONE_{PKT\_time} is the time needed to transmit a _TONE_{PKT}, that is the packet that act as a tone send to the physical layer according to tone-lohi scheme, _curr\_contenders is the number of nodes that participate to the contention and _r_v is a value chosen uniformly at random in [0, 1].

### 2.5 UW-Polling

UW-Polling is a MAC protocol that allows an AUV to collect data from a set of fixed sensors via a polling mechanism. Such a protocol is useful for several reasons. For example, when underwater nodes are sparsely deployed, multihop communications may not be possible, hence a mobile AUV would be required to retrieve data from otherwise disconnected sensors. Secondly, in denser networks, data retrieval may be impaired by concurrent channel access and interference if the amount of generated data is very high. Finally, the AUV may be interested only in a subset of the available data (e.g., the most recent sensor readings, or perhaps only those related to a particular quantity, or those that exceed a certain threshold. UW-Polling may help in all cases above. In particular, the polling mechanism would allow to avoid multiple-access interference; moreover, the AUV can choose the polling sequence and of the data to be transmitted so that the data of higher interest are retrieved first.

UW-Polling has been designed in the context of the CLAM project to address some possible events that may take place in an oil well area monitoring network (which is the main focus in CLAM). For example, the normal operations of the network may include only sensing, and the transfer of sensed data to a sink (connected to a control center) over multihop paths. This does not normally include the presence of an AUV. However one of the following two events may take place:
the readings of the sensors are found to be above some safety levels; the control center ashore or on board the oil rig dispatches an AUV to investigate; the AUV wants to retrieve only some specific data from the sensors that are useful for a better investigation of the abnormal readings received by the sink;

- the data flow at the sink is interrupted for unknown reasons; the control center dispatches an AUV to investigate, and to work as a communications bridge if the interruption is due to network partitioning (e.g., because of the malfunctioning of nodes located along critical multihop paths, or simply because of the exhaustion of their battery).

In both cases, the UW-Polling protocol will help ensure a fast and effective data retrieval process.

We proceed by providing a detailed description of the protocol. UW-Polling works in subsequent phases

- neighbor discovery;
- retrieval of a summary of available data from all neighbors;
- sequential polling of the nodes.

We assume that the AUV is already in the proximity of the underwater sensor network, and that the nodes, in the absence of a multihop path towards the sink, buffer data packets according to some buffering policy (e.g., all generated packets are stored in the queue if it is not full, and only the newest data are kept when there is no more space left).

As a baseline behavior, the AUV will periodically send TRIGGER messages that start the neighbor discovery phase. These messages will go unanswered until some node is found within the communications range of the AUV. Each node that receives the TRIGGER correctly and has buffered data to transmit to the AUV, randomly picks a backoff time, and answers the TRIGGER with a PROBE packet. The backoff is chosen uniformly at random in an interval whose boundaries are communicated by the AUV using the TRIGGER (this eventually leaves some freedom to the AUV to, e.g., reduce the backoff interval if the number of nodes that answers is typically small, or otherwise to extend it).

The PROBE packet contains the information required by the AUV to make decisions on the polling sequence. For example, in case the AUV wants to give priority to the newest generated data, the PROBE should contain the backoff timer chosen by the node (useful for calculating the RTT between the AUV and the node), the number of the packets to transmit to the AUV and the timestamp of the most recent data generated. The number of the packets that a node can transmit to the AUV depends on how many packets a node has in his queue: in practice, this value is limited to some maximum value set as a design parameter, so that multiple nodes can be served in a polling cycle. We assume that this value is preset and known to all nodes. At the end of a fixed time reserved to the reception of the PROBE messages, the AUV assigns a priority to each node from which it received a PROBE. The priority is based on the most recent data generated (e.g., the node that has the most recent data generated in the network will transmit first).
At this point, the AUV transmits the POLL packet which indicates the node to be polled and contains a list of nodes that will be polled in the future, along with the time before their turn will come. The entire polling list is transmitted, so that a node can decide to sleep until its own turn. Upon receiving the POLL, a node checks if it is its turn to transmit; in this case it starts transmitting packets until all those in queue have been transmitted, or until the preset maximum number of transmissions has been reached. In any event, the AUV will set a reception time equal to an RTT, plus the time required to transmit all data packets plus a guard time, which allows the nodes to compensate for the movement of the AUV.

At any of these two points, the turn of the node is completed, and the AUV trims the first element from the list of nodes to be polled, and transmits the new list via a new POLL, which solicits data packet transmissions from the next node in the list. In any event, a polling cycle can only host a given maximum number of nodes, and each node cannot transmit more than a given maximum number of packets. This limits the duration of the polling cycle, in order to ensure that the AUV is not moved too far from the last nodes in the polling list.

Figs. 2 and 3 detail the operations of the UW-Polling protocol for the AUV and the nodes, respectively, by means of the finite state machines that have been implemented.

We describe now in more detail the structure of the header of the packets.

- **TRIGGER**
  - Min_time: the minimum value of the random backoff to be chosen by polled nodes
  - Max_time: the maximum value of the random backoff to be chosen by polled nodes

- **PROBE**
  - Backoff_time: The backoff time chosen by the node, required by the AUV in order to compute the RTT between itself and the node (by measuring the time period from the transmission of the TRIGGER to when the PROBE is received, and subtracting the backoff time value)
  - Npkts2transmit: The number of packets that a node will send to the AUV upon being polled. This value is required by the AUV to compute the time when the nodes in the polling list will be required to start transmitting. This timer also helps avoid that the AUV waits indefinitely for packets from a polled node in case the connection with the node is lost for any reasons. This timer is set to \( T_{data} \times N_{pkts2transmit} \) plus the RTT plus a guard time, where \( T_{data} \) is the data packet transmission time. We note that this timer cannot be very tight, because it must compensate for the movement of the AUV, i.e., with the time-varying RTT.

---

2 As we explain in Section 4, the features of the modem to be employed in the CLAM experiments have been reproduced in the simulations of the present deliverable. This modem does not include a sleep mode, hence the sleeping feature is presently unused.
Fig. 2: UW-Polling: AUV state machine.

Fig. 3: UW-Polling: node state machine.
Deliverable D4.2

- **ts**: Timestamp of the most recent data generated by the node. In the implementation considered in this deliverable, the AUV uses this value to sort the nodes in the polling sequence. Each time a node receives a TRIGGER, it picks the most recent packet from his queue and writes the timestamp of the packet in this field.

- **Id_node**: This value identifies the node which sent the PROBE.

**POLL**

- **n_poled_node**: The number of nodes that are going to be polled
- **POLL[n_poled_node]**: Each element of this vector is a pair of values:
  - **Id_node**: The ID of the node that is going to be polled. When a node receives a POLL, it checks whether the first ID equal to its own ID and, in that case, it starts transmitting data packets. Otherwise it checks if its ID is in the list and waits until it is its turn to transmit.\(^3\)
  - **Waiting time**: The time that a node has to wait before being polled (the first time is always set to zero).

### 2.5.1 Related work

A first evaluation of the performance of random access protocols for the upload of data to an AUV was performed in [6], where the authors compared random access and handshake-based communications in the data retrieval scenario described above.

In [7] the authors compare optical and acoustic communications in a scenario with one mobile node and a field of sensors. In particular, the authors observe that the optical channel can be exploited for high-rate low-range communications whenever possible, whereas the acoustic channel can be used in order to transmit brief control signals over longer ranges. No discussion on networking protocols is given in the paper, as a very simple MAC protocol is employed. In [8] the authors use the acoustic channel to perform neighbor discovery. In particular, the AUV locates static nodes using the vision capabilities enabled by a downward-oriented camera, and then hovers above the static nodes. In addition, the AUV can change trajectory based on the information retrieved from the nodes. While the full autonomy of the AUV (which is initially unaware of the position of the nodes) is a remarkable contribution in the paper, there is still no focus on networking protocols.

In [9], instead, the authors focus on a MAC protocol, describing a Token Polling MAC protocol for Underwater Networks. However, the protocol described works on a two-tiered network where a set of mobile nodes organized in a cluster retrieve data from sensors deployed on the seafloor. These nodes, called Slave Nodes, transmit data to a master node using a polling algorithm. There is one master for each cluster. Furthermore, each Slave node can travel from a cluster to any other. Beyond the

\(^3\) If the modem is sleep-capable, the node can go to sleep at this point, and wake up a moment before receiving the POLL message addressed to itself.
effectiveness of the protocol proposed, the architecture of the network is much more complicated than the one proposed in CLAM.

In [10] the authors describe a MAC protocol (RMAC-M) that, for some aspects, is similar to Uw-Polling. However, there are some assumption that makes this protocol unrealizable in our scenarios. The scenario proposed by the authors, in fact, provides a gateway node that collects all the data packets coming from the sensors. Then, the AUV communicates only with the gateway to collect the data, following a scheme similar to the one adopted in Uw-Polling. In our case, however, the sensors can be far away each other, so it may be difficult to have a gateway that collects the data from the sensors because some sensors may not be in coverage with the gateway. A possible solution is to divide the network into ‘clusters’ and use multiple gateways, but this solution would complicate the network deployment a lot. Our protocol, instead, makes it possible to retrieve reliably the data packets from the sensors using the AUV, even if the sensors are very far apart. Another difference between Uw-Polling and RMAC-M is that the data communication is bi-directional because also the AUV can transmit data to the gateway (i.e., control packets for the sensors, for giving any command such as choose the sensor to use or switch off a part of network).

Taking a look to the wireless radio world, in [11] the authors describe an effective extension of a MAC Polling protocol for IEEE 802.11e standard.

In [12], the authors show (using wireless sensors and a robot) that having an AUV patrol the network and retrieve data from the sensors significantly increases the network lifetime, as sensor nodes can save energy they would employ for long-range communications otherwise.

Finally, in [13] the authors also consider the retrieval of data from a network of fixed nodes using an AUV, and devise a scheduling approach based on Time-Division-Multiple-Access with Acknowledgements (TDMA-ACK) to administer communications. This approach is interesting and achieves good data retrieval performance. However, it requires node synchronization: this can be difficult to realize in practice, and would anyways require some signaling to maintain the common time reference. With the protocol proposed in this paper, we aim at achieving good data transfer performance with no requirements for synchronization or for knowledge of the node positions.

For the purposes of the CLAM project, we need a simple protocol to allow an AUV to retrieve data from a generic network of static nodes. The protocol should be designed explicitly for typical CLAM scenarios and provide good performance in these scenarios: these features are offered by the design of UW-Polling.

3 Routing protocol descriptions

This section presents the design of two routing protocols for CLAM: SUN and CARP.
Our approach in CLAM is to design and refine two different solutions: one based on source routing (SUN), and a second one based on a hop-by-hop relay selection process (CARP). Other than their specific concept, they differ mainly in terms of how the overhead is managed: source routing (hence SUN) concentrates the overhead during the relay selection process, but then the established routes can be used with no need to elect relays at every hop: hop-by-hop relay selection finds relays dynamically as a packet flows towards the destination, which yields advantages in the presence of channels with fast dynamics, but requires to spend a certain overhead cost at every hop. The presence of both approaches in CLAM allows us to cover a wide range of possible scenarios and channel dynamics, and to make quantitative comparisons between the two protocols.

### 3.1 The Source routing for Underwater Networks (SUN) protocol

#### 3.1.1 Protocol Overview

SUN is a reactive source routing protocol. The main idea behind it is that the underwater transmission bandwidth is very narrow compared to terrestrial wireless networks and, usually, it is a good idea to save it for data, and not to waste it with proactive route discovery. This also relates to reducing the energy consumption, as building paths on demand may allow the nodes to save energy.

SUN is reactive since routes are searched for using a Path Request packet when a node has data packet to transmit and knows no route towards the destination (or sink). The Path Request will be propagated through the network in a way that avoids full flooding. At the end of the process, if the node receives at least one answer it will start to send packets to the closest sink following the best path (chosen according to a user-specified routing metric). The routes found this way can be reused for a given time period in order to transmit other data generated later than the packet triggering the route discovery. The expiration time can also be defined by the user and depends on the frequency of topology changes foreseen in the network. For example, in a static network the main source of changes in the status of the links is the variation of the environmental parameters over time. As these parameters tend to vary over a time scale that is usually macroscopic with respect to the packet transmission times and to the packet generation periods, the route expiry periods can be set to some high value, on the order of several minutes.

Moreover, SUN is a source routing protocol because a source node embed a full description of the route to be followed towards the destination in the header of the packet, thereby avoiding hop-by-hop relay election processes.

SUN separately addresses the behavior of two entities: sinks and nodes. Sinks are passive entities because their only tasks in SUN are:

- to periodically send probe messages in broadcast, in order to communicate their presence to the nodes within their communications range;
- to receive data packets.
Instead, all other nodes can:

- send data;
- ask for paths;
- answer path requests;
- act as relays;
- notify route problems such as the loss of contact with a sink or with a given relay.

### 3.1.2 The SUN algorithm

The routing mechanism in SUN works as follows: every sink periodically sends a probe advertisement; if a node receives the message, it understands to be a one-hop neighbor of the sink. These nodes, termed end nodes in the following, will take care to answer path requests and to relay packets directed to the sink. A node considers itself an end node only for a pre-defined period; if in this period it does not receive any further probe from the sink, it will assume that the channel towards the sink is not reliable any more and will release the role of end node. This ensures that end nodes are elected and updated in a way that is consistent with the sink movements.

In addition, there is always a control on the SNR of the sink probe. Namely, the sink probe is “accepted,” (i.e., a node qualifies itself as an end node) only if the SNR of the probe exceeds a certain threshold. This threshold ensures that a transmission to the sink can still be successful for a sufficiently long time, as much as required to cover the time after which a node releases the role of end node.

When the routing module receives a packet from the upper layers, it stores the packet in a buffer. If the buffer is full, the node will drop the new packets until at least one slot of the buffer becomes free.

An agent checks the buffer periodically, and if any packets are found, they are served according to a First-In-First-Out (FIFO) policy. The behavior of the agent is different according with the hop count of the node:

- end nodes (hop count equal to 1) are directly connected to the sink. Hence the agent initializes the packet, in case it was not previously initialized, and sends it to the lower layers (the “initialization” is the process of creating the packet header, and involves the specification, among other fields, of the full sequence of hops that the packet should traverse in order to reach the sink: for an end node, this means sending the packet directly to the sink);

---

4 This is true if the channel can be assumed to be symmetric. Such a condition is not necessarily observed in reality; however, the quality of the return channel from the node to the sink will likely be good also in practice, if the nodes discard the probes received with a signal-to-noise ratio below some threshold. For example, a threshold of 15 dB ensured a return channel with good quality in all simulations.

5 The period is tunable by the user, and it is based on the movement speed of the sink.
• nodes with hop count greater than 1 have a valid path to the sink, which can be reached via multihop relaying. The mechanism is the same of the previous point: the agent initializes the packet and sends it to the lower layer;
• nodes with a hop count of 0, do not have a valid path to the sink. In that case the node leaves the packet in the buffer and starts the Path Establishment process (described later).

We note that fresh initializations are performed as the packet is read from the buffer, in order to employ the latest available path.

When a node sends a Data Packet to the lower layers, additionally:
• it waits for a confirmation of correct reception in the form of a Status Packet and correspondingly starts a Status Packet timeout;
• it increases the number of transmission attempts for this packet.

The Status Packets, therefore, implement a form of cross-layer acknowledgement mechanism. This is required in SUN as the protocol was conceived with generality in mind: at the design stage, it was decided not to assume that error control was provided by any lower layer. This brings several important advantages: i) Status Packets, as confirmations of correct reception by the next hop, can be straightforwardly employed to identify broken routes, in that the absence of Status Packet receptions is a symptom that the previously known route does not exist any more; ii) it is not necessary to operate SUN only on top of link-level protocols that include a form of error control; iii) it is not necessary to make sure that link-level error control avoids retransmitting any control packets and that it provides to SUN the ID of every data packet for which an acknowledgement has been received. By embedding these functionalities within SUN, the protocol takes on some lower-level responsibilities in a cross-layer fashion. However, this design provides several advantages in terms of efficiency and generality.

If a node receives a Status Packet related to the first packet in the buffer, the node will assume that the Status Packet sender correctly received the data packet, and will remove it from the buffer. For each packet a maximum number of retransmissions can be set. If this number is reached, the current node will consider the link to the next hop (read from the packet header) unreliable, and:
• it will create a Route Error packet in order to notify to the source of the packet that the path it used is not available any more;
• if the routing information contained in the header of the packet are the same contained in the routing table of the current node, the node will also consider its routing entry expired and will clear it;
• it will remove the packet from the buffer.

The acknowledgment mechanism used in SUN does not aim to guarantee 100% reliability at the link-level: the maximum number of retransmissions is chosen in order to strike a balance between end-to-end reliability and the need to transmit fresh data, instead of indefinitely retransmitting old data packets. This policy, considered that data packets are transmitted only in the presence of a valid route known by
the packet source, is used to balance between reliability and other metrics such as throughput.

### 3.1.3 Path Establishment Algorithm

As said before, in SUN a node checks if it has a valid route to the sink before transmitting a data packet, and starts a path search process by transmitting a *Path Establishment* packet if no valid route is known. The *Path Establishment* packet will have an option field set to *Request*. When a node receives a *Path Establishment* packet with the option field set as *Request*, it checks if it has already processed a similar packet; if so it drops the packet in order to avoid flooding; if not it adds its own ID to the header of the packet and retransmits it. In this way, a list of valid hops is automatically created on the way towards the destination. This process continues until the packet reaches a node with hop count 1 (an end node). The node with hop count 1 knows that it can reach the sink directly. Therefore, it changes the option field in the *Path Establishment* packet to *Answer* and it sends back to the source via the same hop list contained in the header. Additionally, upon forwarding the answer, every relay node reads the header of the packet and records a valid path to the sink in its own routing table.

If a path to the sink already exists, the source at some point will receive the first *Path Establishment Answer*. This is probably the fastest path to the sink, and the node can use it as a path to send data. The first path received is likely the one with the lowest number of hops, but that does not mean that it is the best one. Hence, the node continues to listen to the channel for other answers. If a later incoming answer describe a “better” packet (according to some metric of interest) the node has an option to update the route. When a node with hop count 1 receives a *Data Packet*, it forwards it directly to the sink.

### 3.1.4 Discussion on Routing Metrics

Routing metrics are used to pick the best path to reach the destination. According to the algorithm above, once a node receives one or more answers, it has to decide which the path through which the data packets should be transmitted. This decision is made using routing metrics.

SUN supports any type of metric that can be implemented by performing local computations at the receiver of an answer or by employing the information written in the header of the *Path Establishment* Answer. By default, two metrics are implemented: the *Lowest Hop Count* and the *MaxMin SNR*. In addition to these two metrics, SUN has a system that allows the user to develop new metrics very easily.

**Lowest Hop Count**—As the name suggests, when a node receives a list of paths, it will choose the one with the lowest number of hops. If two or more paths have the same minimum number of hops, the node will keep the most recent path.
• **Pros**: the algorithm is very simple, and the nodes do not need any additional information. In the packet header, when a node processes a path request, it simply increases the metric field by one, hence the header is also very compact.

• **Cons**: hop count minimization usually means link distance maximization, with a consequent decrease of the average SNR per link.

**MaxMin SNR**—In this case, the header of Path Establishment packets contains a record of the minimum link SNR encountered along the path to an end node. This field is initially set to \(+\infty\) from the source. When a node receives a path request to process, it estimates the receiver-side SNR and updates the field accordingly. At the end, the metric field will contain the lowest SNR in the path between the source and a node with hop count 1. When the source receives a *Path Establishment - Answer* packet, it will update its routing table if the minimum SNR recorded in the packet header is greater than the minimum SNR of the path currently known.

• **Pros**: more reliable paths are typically chosen;

• **Cons**: a slight increase in the size of the Path Establishment packets (hence slightly greater overhead) and a slightly higher processing delay to update the minimum SNR over the path.

After some preliminary simulations, we have chosen to employ the Lowest Hop Count metric in this deliverable. The main reason behind this choice is that the MaxMin SNR metric finds longer routes, as the (shorter) links over these routes typically feature a higher SNR. The greater number of multihop transmissions that ensues increases the amount of traffic injected in the network, especially in the presence of large packet sizes and high packet generation rates. The lower number of transmissions provided by the hop count metric, instead, mitigates this situation.

### 3.1.5 SUN protocol details

Four new packets were defined:

• a *Path Establishment Packet*;

• a *Probe Packet* used by the sinks to notify their presence;

• a *Status Packet* used by the nodes to confirm that packets are correctly received, and therefore identify broken links via lacking *Status Packets*;

• a *Data Packet* that contains the data to be transmitted.

![Fig. 4: Status Packet Header](image)

The *Probe Packet* header is empty. Note that an IP field would be redundant, as the same information is already contained in the IP header.
The Status Packet header (Fig. 4) contains the ID of the packet for which the correct reception is being confirmed. This also informs the data packet sender that the link between itself and the next hop is still active.

The Path Establishment header (Fig. 5) contains a packet type Option that can be Search or Answer (used to define the “direction” of the message), the Metric used by the metric algorithm). The list of hops is initially empty. When a node receives a Path Establishment packet, it checks if its IP is already contained in the list: if so, it drops the packet in order to avoid loops, otherwise it adds its own IP to the list.\(^6\) By performing this mechanism, an end-node that will receive a Path Establishment packet with the option field set to Search can convert directly the packet to an Answer and use the populated list to deliver the answer back to the source of the request.

We remark that the Route Error packet is a special case of the Path Establishment–Answer where the Option field is set to a predefined value: this way, when the packet travels the route upstream to the source, it notifies all nodes that a broken route has been detected, and that fresh path establishment procedures should be carried out.

The Data Packet header (Fig. 6) contains a list of hops to be traversed to reach the sink. Its payload is made of data from the upper layers.

The SUN routing protocol is based mainly on four procedures:

- Algorithm 1 processes a packet when it is received by the node;
- Algorithm 2 is used when the node processes a path search packet;
- Algorithm 3 the third one is similar to Algorithm 2, but refers to the processing of Path Establishment Answers;
- Algorithm 4 manages the buffer of a node and the number of retransmissions performed for each packet.

The pseudo-code of these algorithms is provided below.

---

\(^6\) The maximum number of IPs in the header of a Path Establishment packet is fixed, and the node can add its IP only if there is a free slot.
Algorithm 1 Receive Packet
1: if packet.type = probe then
2: hopcount ← 1
3: else if packet.type = path.request then
4: if packet.option = search then
5: process path search;
6: else if packet.option = answer then
7: process path answer;
8: end if
9: else if packet.type = data then
10: if packet.source = me then
11: if buffer empty = true then
12: drop the packet;
13: else
14: store the packet;
15: end if
16: else if packet.next_hops = me then
17: if buffer empty = true then
18: drop the packet;
19: else
20: store the packet;
21: end if
22: else
23: drop the packet;
24: end if
25: end if

Algorithm 2 Process Path Search
1: if already processed then
2: drop packet;
3: else
4: if hop count = 1 then
5: set packet.option to answer and send back the packet;
6: else
7: add my IP in the header;
8: send the packet in broadcast;
9: end if
10: end if

Algorithm 3 Process Path Answer
1: if packet.destination = me then
2: evaluate path;
3: else
4: if my IP is in the header then
5: evaluate path;
6: forward back the packet;
7: else
8: drop the packet;
9: end if
10: end if
Algorithm 4 Buffer Manager

1: if buffer\_empty = true then
2: schedule another check at a later time;
3: else
4: if hop\_count = 1 then
5: if packet\_already\_initialized = false then
6: initialize the packet;
7: end if
8: send the packet directly to the sink;
9: increment the number of retransmissions of this packet;
10: wait for a Status Packet;
11: else if hop\_count > 1 then
12: if packet\_already\_initialized = false then
13: initialize the packet;
14: end if
15: send the packet to the next hop;
16: increment the number of retransmissions of this packet;
17: wait for a Status Packet;
18: else if hop\_count = 0 then
19: if packet\_already\_initialized = false then
20: send a path request;
21: increment the number of retransmissions of this packet;
22: else
23: send the packet to the next hop;
24: increment the number of retransmissions of this packet;
25: wait for a Status Packet;
26: end if
27: end if
28: end if

3.2 SUN vs DSR

SUN is based on the dynamic source routing paradigm: the routes are dynamic, and the paths are contained in the header of each packet.

The source routing approach has mainly the following advantages:

- there is no need to send periodically in the network hello packets to notify the state of the network;
- it is straightforward to avoid loops in paths;
- routes are created only when required, that means that there is no need to maintain valid an unused route;
- advanced caching systems can be used to exploit implicit information contained in paths required by other nodes;
- each node can decide by himself the entire path to reach a destination according to an internal and personalized metric, regardless the choice of other nodes.

SUN inherits some of the features of DSR [14], adapting and tuning them to underwater scenarios. Some aspects of these protocols are therefore similar. For example: the network organizes and configures itself autonomously; the Route Discovery
mechanism in DSR is similar to the Path Establishment mechanism in SUN; also in SUN there is an option to limit the rate at which the route discovery can be sent by a node; the Route Maintenance mechanism is similar between SUN and DSR; SUN has a Caching Overheard Routing Information mechanism that enables the node to acquire and update their routes easily and just by listening for packets from other nodes; in SUN there are mechanisms to avoid the flooding of Route Request packets.

There are also important differences between SUN and DSR. First of all, in SUN a very important role is taken by end nodes, i.e., nodes that periodically receive Probe packets from a sink. The end nodes feature provides two main advantages: i) end nodes are offloaded the task to answer Path Establishment – Search packets on behalf of the sink, which improves the response time of the network and decreases the chance that routing control procedures cause congestion at the sink; ii) the network topology automatically adapts to changes caused by the mobility of any nodes. Both advantages are key to improving the effectiveness of underwater communications, and only require that the sink periodically sends Probe packets, a feature that is not present in DSR.

In DSR if a node receives a Route Request for which it is not the target, it searches the target of the request in its own Route Cache. If found, the node generally returns a Route Reply to the initiator, rather than forwarding the Route Request. In SUN this is not performed in order to avoid the reuse of old paths. Intuitively, this is a better policy in an underwater scenario where the ratio between the speed of nodes and the propagation speed of the signal is greater than the one for terrestrial wireless networks (scenario for which DSR was originally developed), hence old paths may be no longer valid when a fresh Path Establishment – Request is received. For the same reason, in SUN a maximum validity time for each entry in the routing table can be set. This forces the nodes to periodically refresh their paths if new packets must be sent after the route has expired.

In DSR a node can cache multiple routes to any destination. SUN has been designed for networks with multiple sinks, but keeps trace of at most one route per known sink. Furthermore, we recall that in SUN a node only processes a packet if its IP is written in the header of the packet, and no Overheard Routing Information is written. This is different than in DSR, which stores overheard routing information for later use; however, in doing so, DSR assumes bi-directional links, which are typically an unfeasible assumption underwater.

Like DSR, SUN also has a Route Error mechanism to notify broken routes to packet sources. A node, however, only sends a Route Error regarding a given next hop to a source if that source keeps sending packets that should go through the same next hop. This is done to avoid useless Route Error transmissions.

SUN supports the implementation of different metrics to choose a path, whereas DSR uses only the number of nodes within a path.

SUN implements a buffering system. If the path contained in a buffered packet expires, the node that buffered the packet can update the route contained in the header with a new and valid one.
3.3 Channel Aware Routing Protocol (CARP): a cross-layer approach for underwater routing

3.3.1 Protocol overview

The Channel-Aware Routing Protocol (CARP) is a new distributed cross-layer solution for UWSNs for multi-hop delivery of data to the sink. Next hop selection takes explicitly into account the history of data packet delivery, the link quality and how successful a neighbor has been in forwarding data towards sink. Information about a node residual energy, its storage capacity and its ability to access links towards the sink is collected and used for relay selection. When the network is deployed, CARP starts with a setup phase where each node estimates the number of hops needed to reach the sink \( \text{hop count} \). After the setup phase, the protocol and the information gathered by the nodes start evolving with the packets flowing into the network. When a node has a data packet to transmit, it broadcasts a PING packet carrying the node depth and the number of data packets it has to transmit (length of the data train). It then waits to receive a reply from its neighbors. Upon receiving a PING packet, a node immediately replies with a PONG packet containing: i) its residual energy; ii) the number of data packets that it can stored in its buffer; iii) its hop count; iv) the node “goodness,” which is a measure on how good the node is to forward data to the sink according to link quality information. The selection of the next hop relay is performed at the node that sent the PING packet based on a combination of the information carried by the received PONG packets. The data train is then sent to the selected relay, which confirms its reception sending a cumulative ACK packet. CARP is a cross-layer solution and all the ACK packets are transmitted at the MAC level, no other ACK packets are assumed while the network is operating.

3.3.2 Protocol description

At network set up, HELLO packets are flooded from the sink through the network. In this way, every node \( x \) acquires its hop count \( HC(x) \), i.e., its distance, in hops, from the sink. Each HELLO packet carries information on its source node and the hop count information. The sink generates the first HELLO packet setting its hop count field to 0 and broadcast it to its one hop neighbors. Each node \( x \) that receives an HELLO packet checks whether its \( HC(x) \) is greater than the hop count carried by the packet plus 1. If this is the case, \( x \) updates its hop count to the value in the HELLO packet plus 1, and re-transmits the HELLO message increasing its hop count field by 1. Otherwise, the HELLO packet is dropped. By the end of this flooding process a node has acquired its hop distance from the sink, as well as information about its neighbors towards the sink.
When a node $x$ has one or more data packets to forward it chooses a suitable relay among its neighbors. The search for the relay is initiated by $x$ by broadcasting a control packet, called PING, which carries the following information.

$<\text{src, num_{pkt}}>$.

The field $\text{src}$ is $x$ unique identifier, and $\text{num_{pkt}}$ is the number of data packets that $x$ has to transmit. If $\text{num_{pkt}} > 1$ a train of data packets is transmitted in sequence.

A node $y$ that receives the PING packet immediately replies with a PONG packet which is sent in unicast to the PING source $x$, containing the following information.

$<\text{src, dst, hop, queue, energy, lq}>$.

The fields $\text{src}$ and $\text{dst}$ contain the identifiers of nodes $y$ and $x$, respectively. The field $\text{queue}$ indicates the available buffer space at $y$, i.e., the number of data packets that $y$ can store in its incoming data queue. The field $\text{hop}$ contains $HC(y)$. The parameter $\text{energy}$ indicates the residual energy available at node $y$. The parameter $\text{lq}$ is an indication of the quality of the links outgoing from $y$ (see detailed description below).

Relay selection happens as follows. After sending a PING packet, node $x$ awaits for PONG replies for a time $\delta$. The waiting time $\delta$ is initially set depending on the modem nominal transmission range and on the acoustic signal propagation speed in water. It is then continuously updated by using the actual round trip time of PING/PONG handshakes. After the time $\delta$, node $x$ uses the link quality information $\text{lq}_y$ sent in the PONG messages from all its available neighbors $y$, and combines it with the quality of the link from $x$ to $y$, $\text{lq}_{x,y}$. In particular, for each responding $y$, node $x$ computes:

$$\text{goodness}_y = \frac{\text{lq}_y}{\text{lq}_{x,y}}.$$

The node $y$ with the highest ratio $\frac{\text{goodness}_y}{HC(y)}$ is chosen as the relay, and the (train of) data packet(s) is sent directly to it. In so doing, nodes with a lower hop distance from the sink are preferred. Nodes with a higher hop count are chosen only if their link quality is significantly better than those closer to the sink. If there are ties, priority is given to the node with the highest energy, and then to the node with the higher available buffer space (as encoded by $\text{queue}$). Further ties are broken by using the node unique identifiers. Upon receiving a train of data packets, a node $y$ replies with a cumulative ACK, acknowledging each single packet in the train (bit mask). Upon receiving an acknowledgment from $y$, node $x$ updates its hop count to $HC(y) + 1$. In this way, the hop count information is dynamically updated according to possible changes of the network topology. When node $y$ has received a train of one or more data packets, it checks whether it has received them previously, so to re-transmit only those that it has not forwarded already.

**Computing the link quality $\text{lq}$**

The $\text{goodness}_y$ that each node $x$ computes for all its neighbors $y$ that replied with a PONG represents an estimate of the quality of the channel from $x$ to $y$ and from $y$ to its best reachable neighbor in a route to the sink. The link quality $\text{lq}_y$ is computed by $y$ based on the success of past transmissions to its neighbors. It is defined as
an exponential moving average, where the weight of transmissions back in the past are less influential than more recent ones in assessing the goodness of the link for transmissions. This enables CARP to take into account the time varying nature of the channel, giving more importance to what has happened recently. More formally, for each data packet transmissions $t$ to one of its neighbors $z$, node $y$ computes:

$$lq_t^z = \alpha Y_t^z + (1 - \alpha)lq_{t-1}^z.$$  

The coefficient $\alpha \in (0, 1)$ is the constant smoothing factor through which we can control how quickly the influence of older transmissions decreases. For instance, a higher $\alpha$ could be used for very variable underwater channels, as it discounts older transmissions faster. $Y_t^z$ is the success ratio of the $t$th transmission from $y$ to $z$, defined as the ratio between the number of data packets correctly received by $z$ (i.e., that $z$ acknowledges positively) and the number of packets sent in the train of that transmission. $lq_{t-1}^z$ is the value of the moving average after $t - 1$ transmissions from $y$ to $z$. Since this definition is recursive, we define $lq_1^z$ as the success ratio of the first transmission. The value $lq$ that node $y$ transmits in its PONG message to $x$ is the best among the $lqs$ to all its neighbors, based on the (different) data transmissions with each of them. The value $lq_{x,y}$ used by $x$ for computing goodness is computed similarly, considering data packet transmissions from $x$ to $y$.

The goodness value estimation does not assume period packet transmission to estimate how the link quality changes over time. This estimation is based only on the different packets transmitted and received in the network to actually reserve the channel and transmit the data. In case of a medium/high traffic load, several packets are transmitted in the network and the goodness value can be easily update over time by the different nodes to estimate the quality of the links in the network. If the traffic load is low, a reduced number of data and control packets are generated in the network and a less precise information about the link quality is estimated. On the other hand, however, if the generation rate of the data packets to be delivered in the network is low, periodic control packet transmissions for goodness estimation will result in consuming more energy for the link quality estimation than for the actual data transmission in the network. In fact, since the underwater channel is highly dynamic, the period time for control packets transmission to estimate link quality should be quite short strongly increasing the energy consumption of the nodes in the network.

Transmission power considerations —

Data packet transmission happens at a higher power than that used for sending PING and PONG packets. This is done to keep the same PER (packet error rate) for both data and control packets. Data packets are usually longer than control ones. If a relay node is selected based on the correct transmission of short control packets, it is then possible that the same link will be really unreliable when long data packets want to be transmitted. Therefore, CARP makes it possible for data packets from $x$ to $y$ to have the same probability of success than that of PING packets from $x$ to $y$ by gauging its transmission power appropriately (as to say: “If I can contact you, I can also talk to you”). More precisely, the transmission power $P_1$ of control
packets and the transmission power $P_2$ for data packets are computed as follows. Using a BPSK modulation the probability to transmit correctly a packet that is $n$ bits long is $Pr$(correct transmission of $n$ bits) = $(1 - BER)^n$. The BER is computed as $BER = \frac{1}{2}\text{erfc}(\sqrt{SNR})$, where erfc() is the complementary error function. The $SNR$ is given by $\frac{P/A(l,f)}{N(f)\Delta f}$, where $P$ is the transmission power, $A(l,f)$ is the attenuation in the underwater channel over a distance $l$ for a signal of frequency $f$, $N(f)$ is the noise p.s.d. and $\Delta f$ is the receiver noise bandwidth [15]. If control packets are $n_1$ bits long and data packets are $n_2$ bits long, we want to determine $P_1$ and $P_2$ so that $Pr$(correct transmission of $n_1$ bits) = $Pr$(correct transmission of $n_2$ bits). Therefore:

$$
(1 - \left(\frac{1}{2}\text{erfc}\left(\sqrt{\frac{P_1}{A(l,f)N(f)\Delta f}}\right)\right)^{n_1} = (1 - \left(\frac{1}{2}\text{erfc}\left(\sqrt{\frac{P_2}{A(l,f)N(f)\Delta f}}\right)\right)^{n_2}
$$

$$
(1 - \left(\frac{1}{2}\text{erfc}\left(\sqrt{\frac{P_1}{A(l,f)N(f)\Delta f}}\right)\right)^{n_1} = (1 - \left(\frac{1}{2}\text{erfc}\left(\sqrt{\frac{P_2}{A(l,f)N(f)\Delta f}}\right)\right)^{n_2}
$$

$$
\left(\frac{1}{2}\text{erfc}\left(\sqrt{\frac{P_1}{A(l,f)N(f)\Delta f}}\right)\right)^{n_1} = 1 - \left(\frac{1}{2}\text{erfc}\left(\sqrt{\frac{P_2}{A(l,f)N(f)\Delta f}}\right)\right)^{n_2}
$$

$$
\left(\frac{1}{2}\text{erfc}\left(\sqrt{\frac{P_1}{A(l,f)N(f)\Delta f}}\right)\right)^{n_1} = 1 - \left(\frac{1}{2}\text{erfc}\left(\sqrt{\frac{P_2}{A(l,f)N(f)\Delta f}}\right)\right)^{n_2}
$$

$$
P_2 = \text{erfcinv}^2(2 - 2\left(1 - \left(\frac{1}{2}\text{erfc}\left(\sqrt{\frac{P_1}{A(l,f)N(f)\Delta f}}\right)\right)^{n_2}\right)A(l,f)N(f)\Delta f
$$

$$
P_1 = \text{erfcinv}^2(2 - 2\left(1 - \left(\frac{1}{2}\text{erfc}\left(\sqrt{\frac{P_2}{A(l,f)N(f)\Delta f}}\right)\right)^{n_2}\right)A(l,f)N(f)\Delta f
$$

4 Simulation scenarios and parameters

We now present the scenarios used in the following sections to simulate the performance of MAC and routing protocols. First of all, we recall that, in this deliverable, our objective is not to simulate a fully realistic deployment regarding the CLAM target application, i.e., oil well and pipeline monitoring and control. Rather, the objective of this deliverable is to present protocol design principles to be applied to such a scenario: to better present these principles, we have considered simulation scenarios and network topologies that resemble the final application scenario, and yet allow us to put the protocol under stress. In turn, the latter makes it possible to measure certain metrics of interest such as the maximum amount of traffic that can be reliably supported by the protocols, the delay experienced by packet transmissions in this scenario, the expected throughput in conditions of low as well as high traffic, and the spatial distribution of these and other metrics.

In the following, we will consider three different scenarios:
Fig. 7: A scheme of the node deployment in scenario 1. Every box contains the ID of the node randomly placed within the box. The sink is always placed in box 5, 500 m north and 500 m west of the bottom-left corner of the picture, 10 m below the surface, and is represented here as a surface buoy.

- **Scenario 1:** 10-nodes, single-hop (see Fig. 7), employed to simulate the performance of MAC protocols. In this scenario, the network area is square-shaped with limited size (1 km × 1 km), the sink is centrally placed, and the remaining 9 nodes are placed uniformly at random in separate, equal-size portions of the area to be monitored. The scenario has been designed in order to put the MAC protocols under progressive stress. This allows us, on one hand, to study the performance of the MAC protocols under different conditions in terms of traffic and generated packet sizes; on the other hand this study is akin to the simulation of a local “cluster” of nodes, e.g., deployed within an oil drilling area in order to monitor the oil extraction operations.

- **Scenario 2:** 20-nodes, multihop (see Fig. 8), employed to simulate the performance of a more complete protocol stack including a routing protocol. The network area is twice as large as in the previous case (1 km × 2 km), it has a rectangular shape, and forces the packets to be routed through typically 1 or 2 relays before reaching a sink placed at the center of one of the short sides.

- **Scenario 3:** 40-nodes, multihop (see Fig. 9), where the area is now 1 km × 4 km; this scenario increases the network length and requires the packets to travel 4 to 6 hops before reaching the sink. We employ this topology to test the capability of the routing protocols to convey data to a sink through more hops than in Scenario 2.

As can be inferred from the above information, the number of nodes increases proportionally to the network area, hence the average density of the nodes is the same. Of these nodes, one is always the sink, which is assumed to be the final destination of all generated packets. In all scenarios the sink is assumed to be a moored buoy placed 500 m north and 500 m west of the bottom right corner of the network areas depicted in Figs. 7, 8 and 9, on the surface, with the receiving hydrophone hanging from the buoy at a depth of 10 m. In scenario 1, all packets are directly delivered to the sink, as all nodes are within its coverage distance. In scenarios 2 and 3, instead, the area is larger, and therefore only part of the nodes are within
Fig. 8: A scheme of the node deployment in scenario 2. Every box contains the ID of the node randomly placed within the box. The sink is always placed in box 5, 500 m north and 500 m west of the bottom-left corner of the picture, 10 m below the surface, and is represented here as a surface buoy.

Fig. 9: A scheme of the node deployment in scenario 3. Every box contains the ID of the node randomly placed within the box. The sink is always placed in box 5, 500 m north and 500 m west of the bottom-left corner of the picture, 10 m below the surface, and is represented here as a surface buoy.

In collaboration with WP2, we identified three packet sizes that are representative of significant data that could be generated in underwater networks for oil well monitoring:
• **100 bytes**: this size represents monitoring data originated by sensors of different types, e.g., a CO₂ sensor that periodically reports the status of a carbon storage area back to the control center.

• **500 bytes**: this size represents more bulky data generated by specific sensors, e.g., vibration sensors that transmit the main features of recorded audio samples.

• **1000 bytes**: this size is the largest considered in our scenarios, and can represent more detailed reports provided by the same sensors above, or an aggregated set of readings of the same sensor to be transmitted altogether in order to increase the efficiency of handshake-based protocols; alternatively, it can represent an aggregated version of the readings of one or more nodes.

In every simulation and scenario, the nodes generate only one of the packet sizes above.

The values of the packet generation rate per node (denoted with \( \lambda \)) have also been chosen to represent typical reporting periods that would be desired in underwater oil well applications. These values have been derived in collaboration with WP2, and are based on meetings with industrial actors in the oil drilling business as well as on the past experience of the industrial partners in CLAM. In more detail:

• the lowest packet generation rate possible in oil well scenarios is **1 packet from each sensor or node every 6 hours**: this would represent the lowest reporting rate in “typical” scenarios, i.e., where no alarm or no other particular event is detected and every asset is functioning within normal parameters.

• the highest packet generation rate is **1 packet every 5 minutes**. This rate is 72 times higher than the one above and represents instead the wanted reporting rate in conditions of alarm or danger; in these conditions, the network should respond more quickly, so that the control center can gather a more detailed (or more up-to-date) report of the event being sensed.

The set of values of \( \lambda \) in the simulations in Sections 7 and 8 has been chosen to include the above rates, and is independent of the packet size. In other words, we make separate tests for, e.g., one packet of 100 bytes per node every 5 minutes and for 1 packet of 1000 bytes per node every 5 minutes. This choice is not necessarily representative of a real scenario, but allows us to test the protocols under stressful traffic conditions, which is one of the main objectives of this deliverable. For example, having every node in a network of 20 or 40 nodes generate one packet of 1000 bytes every 5 minutes creates heavy traffic and is likely to be unbearable by most protocols. Hence, when we simulate the full set of packet generation rates, we are likely to find the rate yielding, e.g., the maximum network throughput supportable by each protocol, which in turn identifies the packet generation rates for which the protocol behaves well, and those for which the transport capability of the protocol is saturated. In the simulations, the packet arrival process has Poisson statistics with average inter-arrival rate \( \lambda \).

All generated packets are hosted in a queue on the node before being served according to the rules of the protocol in use. The same queue is also employed to temporarily store the packets received from neighboring nodes. The size of the
queue is fixed to 10 kbytes of storage per node, allowing to store 100 100-byte packets, 20 500-byte packets or 10 1000-byte packets.

The parameters of the modem have been chosen to be in line with the specifications of the Kongsberg modem available in CLAM (updated to December 2011). An interaction with WP3 was profitably set up in this regard. A list of parameters follows:

- **Bit rate**: 1000 bps
- **Carrier frequency**: 25.6 kHz
- **Bandwidth**: 4 kHz
- **Source Power Level (SPL) at 1 m**: 178 dB re µPa
- **Transmit / Receive / Idle power**: 3.3 W, 620 mW, 85 mW
- **No sleep mode**
- **Modulation**: BPSK
- **Acquisition threshold**: 1 dB
- **Node battery capacity**: 78 Ah; at an output voltage of 14.4 V, this translates into a nominal maximum of about 4 MJ of available energy.\(^7\)

The simulator provides a measure of the *input* Signal-to-Interference-and-Noise Ratio (SINR) at the receiver. In order to model the fact than only part of the energy carried by the channel can be harvested for signal detection, we applied a fixed penalty of 10 dB to the input SINR. As specified in D5.1, we extended the simulator to track the time-varying probability of error resulting from time-varying interference affecting the reception packet. This extension is included in the results of this deliverable.

We recall that the WOSS package [1] employed for simulation along with with nsMiracle [16] allows to interface nsMiracle to the Bellhop ray tracing software [17], in order to get more realistic realizations of the channel power gain between the a transmitter or interferer and the receiver. In turn, this enables a more realistic computation of the SINR. The Bellhop software requires the specification of some environmental parameters affecting the propagation of sound, in order to compute the ray trace and derive the sound pressure at the receiver. These parameters include the sound speed profile (SSP), i.e., the function that relates sound speed and depth, the geoaoustic parameters of the bottom sediments and the shape of the bottom. Optionally, the altimetry (the shape of the surface) can also be set (see D5.1 for more details). For the simulations in this deliverable, we interacted with WP2 and WP5 in order to model the same environment that would be found in the Trondheim fjord, the location of the final CLAM demonstration. Unfortunately, this fjord is not covered by the standard databases from which WOSS derives environmental data (the World Ocean Database [18], the GEneral Bathymetric Chart of the Oceans [19] and the Deck41 bottom sediments database [20]). Therefore, we had to rely on in-situ measurements. The participation of SINTEF to the EU FP7 UAN [21]

---

\(^7\) KM states that this battery capacity is a conservative estimate, and can hence be used directly. In the following, we will assume that only half of this capacity (2 MJ) is available to the modem, while the rest is required to operate other components of the CLAM node, including the sensors and the embedded computer controlling the modem.
The project was key in this regard, as the final UAN demonstration was also held in the Trondheim fjord. SINTEF contacted the Project Leader of UAN, and obtained the permission for CLAM to reuse the same environmental data gathered during UAN experiments. This includes two SSP realizations in different locations of the fjord and a map of the bathymetry of the area where the experiments will be held. While we refer the reader to the CLAM test plan for more details, we report in Fig. 11 some SSPs measured in the fjord, courtesy of SINTEF and of the UAN project [21]. The closest to the foreseen test area for the final CLAM demonstration is the “Trollet” location at the center of the fjord. Fig. 12 shows a 3D plot of the bathymetry around the same location.

We have found no precise account of the bottom sediments in the CLAM fjord. However, SINTEF’s past experience suggests that the bottom sediments are a mixture of fine clay and mud, with no vegetation. This is consistent with what SINTEF observed on the equipment recovered from the sea floor. As no observation on the spectrum of surface waves is available, we approximate the sea surface as flat in Bellhop simulations.

Environmental noise is modeled using the empirical power spectral density equations reported in [15, 22]. As parameters to those equations, we chose a moderate shipping factor of 0.5 and a wind speed of 7 m/s. The latter corresponds to level 4 of the Beaufort scale, which is typically observed at the Vaernes airport, 17 km east of the test site (Beaufort levels 5 and 6 were very infrequently observed in the period 1970-2011). We refer to the CLAM test plan for a more complete account of wind speeds measured near the Trondheim fjord.
Fig. 11: SSP at three different locations in the Trondheim fjord. The one of interest in this document is the "Trollet" location, which is closest to the area of the final CLAM demonstration.

Fig. 12: Bathymetry profile at the center of the Trondheim fjord, near the area marked by the pentagon at the center of Fig. 10.

Fig. 13: West-east bathymetry slice centered on the white circle in Fig. 12.

Fig. 14: South-north bathymetry slice centered on the white circle in Fig. 12.
In this section we briefly list and define the metrics we extracted after the simulations in order to address the performance of the protocols. For each metric, a unit of measure is reported in square brackets if applicable. We consider Scenario 1 first as it is specifically configured for MAC simulations, and then we proceed to Scenarios 2 and 3.

5.1 Scenario 1

- **Application throughput per node at the sink in [bytes/minute]**, defined as the size of the payload of data packets that correctly reach the sink (i.e., excluding replicated and out-of-order packets), expressed in bytes per minute;

- **Packet delivery ratio (PDR) at the sink**, defined as the ratio of the packets correctly received at the sink divided by the packets generated by the nodes;

- **Packet delivery delay per kilometer [s/km]**, defined as the length of the period of time from the time a packet is generated to the time it is delivered to the sink, divided by the distance of the source node from the sink in kilometers;

- **Energy consumption per node [J]**;

- **Average number of link-level retransmissions per packet**, defined as the number of packets written in the buffer of the nodes by the application layer, divided by the number of packets received at the sink, minus 1. This metric only applies to the versions of the protocols that employ error control.

- **Ratio of the packets dropped for interference**, defined as the ration between the packets dropped for the interference present in the network, and the number of total packets dropped for noise, interference, deafness of the receiver. Only data packets is considered (any type of control message such as RTS/CTS packets and ACK packets is not taken in account). Moreover, the destination of the packet is checked. In this way, each receiver take in account only the packets that are actually for itself and, other packets that are received and are going to be discarded at MAC level are not taken in account at PHY level (where actually this counting is done).

5.2 Scenarios 2 and 3

Scenarios 2 and 3 involve multihop routing. The metrics defined for scenario 1 are still of interest, with the understanding that the way they are defined in Section 5.1 is still valid, and has the effect to turn them into end-to-end metrics for scenarios 2 and 3. For a better analysis of routing protocols, we also include an additional overhead metric. The full list of metrics is as follows
• All metrics listed for scenario 1;
• **Protocol overhead ratio**, defined as the ratio of all control bits to the sum of all control bits and all data bits correctly delivered to the sink. The definition of “control bits” refers to the following:
  – the size of any control packets;
  – the size of the header of any data packets transmitted for any reason (newly generated packets, relayed packets, packets retransmitted because of link errors).

The definition of “data bits” refers instead to the size of the payload of every unique data packet correctly delivered at the sink in order (i.e., the same quantity employed for the computation of the throughput).

• **Normalized energy consumption per node**, defined as the ratio of the total energy spent by a node during a simulation, divided by the energy the node would spend if it stayed idle for the whole simulation time.

The metrics above will be plotted in separate graphs for each protocol, as a function of the packet generation rate per node $\lambda$ expressed in [packets/minute/node]. The values of lambda range from $2.4 \cdot 10^{-3}$ (slightly less than one packet every 6 hours) to 0.2 (1 packet every 5 minutes). This interval includes the values chosen in collaboration with WP2 to be representative of the oil well monitoring application. As we are going to test the protocols with packets of 3 different sizes, the plots will contain three lines, one per packet size. For all simulations, the results will be averaged over 10 different realizations of the network topology specified by the respective scenario. The realizations are drawn such that every node has a path to the sink where the SNR (over each link of the path) is at least 9 dB. In each realization, the simulation time is set to a value that allows all nodes to send about 100 packets, plus a period of 1000 s to allow all nodes to empty their queues. In other words, the amount of simulated time is computed as the inverse of the packet generation rate, plus 1000 s.

Before moving to presenting the results of simulations, we recall from Section 4 that, in every scenario, the network area is subdivided into elements of roughly equal size ($3 \times 3$ elements in scenario 1, $3 \times 6$ elements in scenario 2 and $3 \times 12$ elements in scenario 3), and the nodes are randomly placed within one of these elements. Even in different realizations of these random topologies, a node with the same ID is always placed within the same grid element. This allows to plot a three-dimensional graph to show the spatial distribution of the metrics above. This analysis will be carried out for Scenario 3, and will focus on the throughput, the end-to-end delivery delay and the normalized energy consumption per node.
6 Optimal packet size selection

In this section we introduce the effect of different payload sizes on the performance of MAC protocols and thereby motivate why we performed tests with different payload sizes as introduced in Section 4.

While design of underwater MAC and routing protocols for UWSNs have flourished in the past few years [3, 23–33], only a few analyses are concerned with parameter optimization, and in particular with the performance improvement induced by choosing the best payload size given transmission range, bit rate and the error probability. The importance of optimal payload size selection and its effect on the performance of underwater MAC protocols in single-hop and multi-hop networks has been pointed out in different works [34–36]. In all these works some simplistic assumptions about the network or underwater acoustic channel models have been considered.

In a multi-hop network, the noise and fading-induced BER is not the only cause of packet loss. Here, interference is an important factor that contributes to performance degradation. This situation is exacerbated in acoustic scenarios where the low spreading factor (path loss exponent) supports interference from non-neighboring nodes, and even from those that are far away in the network. Through extensive simulations on most of the underwater MAC protocols proposed so far, it has been observed that the vast majority of packet losses are due to this latter type of interference [27]. Specifically, in the case of RTS/CTS-based access, we observed that 90% of packet losses are due to interference coming from nodes that are outside of the receiver’s transmission range. This effect occurs even in networks where the traffic is not particularly high. Moreover, many of these collisions happen between control and data packets.

Therefore, we have decided to conduct a more complete investigation of the impact of payload size on the performance of multi-hop communications in an underwater network, varying transmission rates and channel BERs and assuming a complete signal interference model. The objective is to determine the payload size that provides the best performance with respect to key metric such as: throughput efficiency (defined as the ratio between the delivered and the offered bit rate); data packet latency (measured “per meter” to unify the performance over varying deployment areas); energy consumed for each bit correctly delivered to the network data collection point (the sink). We have considered two underwater MAC protocols, namely, Carrier Sense Multiple Access (CSMA) and the Distance-Aware Collision Avoidance Protocol (DACAP) [3], which exemplify MAC schemes with and without the RTS/CTS handshake for collision avoidance, respectively. Our choice is motivated by a comparative performance evaluation among several MAC protocols that we performed in [27], that showed these two protocols as among the best performing in the multi-hop scenario.

The multi-hop scenarios we have investigated are challenging in that we consider a relatively large number (100) of nodes randomly deployed over an arbitrary shallow water area, and data generation rates corresponding to different application requirements. We expect this to be the core scenario of future underwater network
deployments, where further components could be added, such as mobile unmanned devices, or support for underwater cellular-like communications. We have also investigated different network sizes (16 and 35), topologies (single-hop and multi-hop) and deployment areas, and discuss how these parameters affect payload size selection. A comprehensive summary of the results can be found in [37]. Results are obtained through ns-2-based simulations [38] combined with the Bellhop ray tracer [17] for modeling the acoustic channel propagation and signal attenuation. The Bellhop ray tracing model is used with real environmental data that provide us with a first approximation of the underwater acoustic channel behavior.

6.1 System Model

CSMA and DACAP have been implemented using nsMiracle [16] on top of ns-2 [38], connected to the Bellhop propagation simulator [17] via the WOSS interface [1]. Bellhop allows us to compute the frequency-dependent acoustic path loss of each source-destination pair at a given location, as well as the spatially-varying interference induced by all active nodes. The ray tracing model is used with real environmental data of an area located in the Mediterranean sea off the coast of the Pianosa island (Tuscan archipelago), with the coordinate (0,0,0) of the surface located at 42°32′0″N and 10°22′0″E. In particular, we used the sound speed profiles, bathymetry profiles and information on the type of bottom sediments of the selected area obtained from the World Ocean Database [18], from the General Bathymetric Chart of the Oceans (GEBCO) [19] and from the National Geophysical Data Center’s Deck41 data-base [20], respectively. The SSP is retrieved by WOSS from the World Ocean Database (average of measurements from September 2009). The acoustic field is obtained through Bellhop ray tracing for a signal source located at a depth of 50m. The bottom type is clay and silt.

6.2 Simulation scenarios and settings

Parameter setting as well as the characteristics of selected topologies are shown in Table 1. We consider networks with 100 nodes (99 nodes plus the sink) statically placed in a region with surface 4km × 4km. Nodes are placed randomly and uniformly at different depths, ranging from 20 to 100m. Every node has an average of 15 neighbors. The sink is placed centrally on the surface with the transducer 10m below. Packets are transmitted from the nodes to the sink through pre-determined shortest routes. Each packet that makes it to the sink traverses an average of 2.5 hops (the maximum number of hops is 4). Topology construction also ensures that the selected routes are made up of robust links (with respect to SNR).

We considered three bandwidths, namely, 200Hz, 2000Hz and 20000Hz. Bandwidth efficiency is set to 1bps/Hz and we assume BPSK modulation. The carrier
frequency is 24kHz for bandwidths of 200Hz and 2000Hz, and 22kHz for the third bandwidth. For each value of the bandwidth we have computed the transmission power $P_{tx}$ (in dB) that results in average BERs on the selected routes equal to $10^{-4}$ or $10^{-6}$. Traffic is generated according to a Poisson process with aggregate (network-wide) rate of $\lambda$ packets per second. Once a data packet is generated, it is associated with a source selected randomly among all nodes. The destination of all data packets is the sink. We define the normalized packet rate as $\lambda_{\text{T pack}}$, whose values are considered in the range 0 to 1 packets per packet time. Packet time is expressed as $T_{\text{pack}} = \frac{N_b}{R_b}$, where $N_b$ is the packet size in bits and $R_b$ is the bit rate. Simulation results presented here concern very low traffic ($\lambda = 0.01$), low traffic ($\lambda = 0.1$), medium traffic ($\lambda = 0.2, 0.3$) and high traffic ($\lambda = 0.6$). Results from simulations with very low traffic are shown only for scenarios where the nodes transmit at 2000bps and 200000bps, while results for high traffic are shown only for scenarios where nodes transmit at lower bit rates. This pairing is made because traffic is normalized, and, as a consequence, the actual number of data packets injected in the network for a given $\lambda$ increases with the bit rate. When $\lambda = 0.6$ and the bit rate is 2000bps or higher, the network becomes congested, and performance is considerably degraded.
In order to assess the impact of the payload size on the protocol performance, we consider data packet payloads of 100B, 200B, 400B, 600B, ..., 2800B, 3000B (for a total of 16 different payload sizes). The total size of a data packet is given by the payload plus the headers added by the different layers (physical through network). The physical layer header contains all the information needed by the modem to correctly start receiving a packet (synchronization preamble, delimiters, etc.). At the physical layer, nodes need a synchronization peering time which is taken to be on the order of 10ms (the physical header overhead changes according to the data rate). The MAC header contains the sender and the destination IDs, and the packet type. Its length is set to 3B. The sizes of RTS and CTS packets are set to 6B, and ACK and WARNING packets are 3B long. Packets are discarded because of channel distortion (BER), collisions and interference. We do not consider packet loss due to malfunctioning hardware. Whenever the buffer is full and a new data packet arrives, the oldest data packet is discarded. Our implementation of CSMA mandates to discard a data packet after 7 attempts of either accessing the channel or re-transmitting the packet. The same holds for DACAP concerning RTS packets. For data packets only 4 attempts of either accessing the channel or re-transmitting are made before the packet is discarded (values tuned through simulations). For both protocols we consider the version with acknowledgments, which proved more robust, especially in multi-hop scenarios [27].

6.3 Performance results

Figure 15 shows the results for the different BERs and bit rates considered.

When a low BER ($10^{-6}$) is considered, for both CSMA and DACAP the optimal payload size steadily increases with the offered load. In fact, making use of short payloads to deliver the same amount of bit to the sink, more data packets have to be transmitted thus increasing the overhead imposed by control packets, as well as the probability of collisions and the number of times the nodes find the channel busy increases.

This is especially true in a multi-hop scenario since each hop generates extra data packets, new overhead (control packets) and collisions also happen because of interference generated by transmitting nodes multiple hops away. The use of RTS and CTS affects DACAP especially for short data packets, and when the propagation delay is overwhelming with respect to the transmission delay, which makes the handshake duration particularly long. For this reason DACAP shows the best performance with longer data packet sizes with respect to CSMA. Not having to endure extra delays for accessing the channel, CSMA instead prefers short data packets when the traffic load is low. At the lowest traffic loads, both protocols are able to deliver all the generated data packet and shorter packet are preferred since they reduce the end-to-end packet latency, especially at lower bit rates.

As the BER increases to $10^{-4}$, the situation changes considerably. Given the high BER, longer packets suffer from a higher probability of being corrupted dur-
ing transmission and therefore require re-transmission. The throughput no longer increases steadily with the payload size: It reaches a maximum and decreases thereafter. The optimal payload size depends on the offered load and on the bit rate. For example, when $\lambda = 0.1$ and $R_b = 2000$bps, the maximum achievable throughput is about 65% for CSMA and 45% for DACAP, which is quite a decrease from the 97% seen when the BER is $10^{-6}$.

6.4 Varying the deployment scenarios

We have also investigated how key parameters, such as the network size $N$, the type of network topology (single-hop vs. multi-hop) and the size of the deployment area affect the optimal data packet size selection.

The first set of simulations refers to a network of 15 nodes scattered uniformly in a 700m × 700m area. Each node can transmit directly to the sink, which is located centrally on the surface (single-hop topology). All other parameters are the same as those described in Section 6.1. The payload sizes that optimize throughput performance for both CSMA and DACAP are shown in Figure 16 for different bit rates $R_b$ and BERs.
A second set of simulations concerns a multi-hop network with 34 nodes, scattered uniformly in an area of 2000m $\times$ 2000m. As before, the sink is placed centrally on the surface. The average route length traveled by data packets from the nodes to the sink is 1.55 hops. Results for this set of simulations are depicted in Figure 17.

In general, the metrics investigated (throughput efficiency, latency per meter, and energy per bit consumption) show similar trends in all considered scenarios ($N = 16$, $N = 35$ and $N = 100$). When the BER is $10^{-6}$, increasing the packet size reduces the overhead, leading to better throughput efficiency. Increasing the network size and the route length increases the network traffic, favoring larger payload sizes for a given offered load. Larger payloads are particularly beneficial for DACAP, because of the control overhead required for channel acquisition. In networks with BER = $10^{-4}$ we observe two contrasting effects: Increasing the payload size reduces the number of contentions; at the same time, however, the higher BER makes it more likely for a larger packet to be discarded because of errors. The combination of these two effects causes the optimal payload size to decrease with the BER.

Despite the fact that trends are similar for different scenarios, the values of payload sizes depicted in Figure 15, Figure 16, and Figure 17 are noticeably different, suggesting that the packet size needs to be carefully tuned to the specific scenario for optimum performance.
In summary the majority of existing acoustic modems are designed to use a priori determined packet sizes. Based on the presented results, “rough guidelines” can be suggested for the design of practical systems, by showing which payload size optimizes throughput efficiency in which scenario. We have compared the performance of CSMA and DACAP, two exemplary MAC protocols for underwater WSNs, with respect to payload size selection in deployment scenarios with non-zero BER and interference, parameters that were not previously considered. We observed that appropriate selection greatly depends on the protocol chosen, and especially on whether it uses control packet extensively (DACAP) or not (CSMA), on the system parameters (bit rate and BER) and on the traffic (packet arrival rate). Results show that CSMA experiences improved performance with shorter payloads, while DACAP, whose collision avoidance is implemented explicitly through a full handshake, shows better performance with long payloads. In scenarios where modems are equipped with low power wake-up capabilities, we observed benefits to energy consumption, especially for DACAP, whose performance becomes similar to that of CSMA, or better with larger payload sizes. These findings have an implication on the design of practical acoustic systems as they point to the fact that choosing a payload size a priori, in an ad hoc manner, may severely penalize the overall throughput performance. For this reason in the following we have varied also the payload size when assessing the performance of the different proposed MAC protocols.
7 MAC protocol simulations

In this section, we present the results of MAC protocol simulations. All simulations are performed using Scenario 1 described in Section 4. For all protocols listed in Section 2 we present both a version without error control and a version employing S&W ARQ.

7.1 CSMA without error control

We start with CSMA without error control. In Fig. 18, we plot the throughput per node as a function of the packet generation rates of interest in CLAM, for the three packet sizes of 100, 500 and 1000 bytes. The curves are almost linear for all packet sizes, meaning that the protocol never operates in the region of saturation. In turn, this fact is mainly due to the very light channel access procedure implemented by CSMA, which is based on short carrier sensing phases with no exchange of data between the nodes.

The linear increase of the throughput curves with traffic is a good index that the network is never congested, i.e., that the data packets generated can be delivered with at least an acceptable probability of error. This fact is confirmed by the plot of the packet delivery ratio in Fig. 19, where the probability that a generated data packet is correctly delivered to the sink never decreases below 0.87 for all packet sizes. We note that using a packet size of 100 bytes allows the network to achieve the highest delivery ratio but the lowest throughput among the three packet sizes. Such result is expected because the network never operates in a congested regime, hence the main cause of packet loss is channel noise, which in turn is more likely to affect longer packets than shorter packets. In turn, pushing more data through the channel via data packets of longer size improves the throughput via a more intense utilization of the medium. As a further reason behind the good throughput in the 1000 m case, we recall that the protocols based on CSMA work best when the packet transmission time is (possibly much) greater than the propagation delay. Using packets of 1000 bytes achieves this condition, as the transmission time of a data packet (8 s at 1000 bps, plus the header transmission time) is sufficiently longer than the maximum propagation delay over the channel (about 1 s from the farthest corner of the network area to the sink). To summarize, with packets of 1000 bytes the efficiency of CSMA improves, at the expense of possible waiting times imposed by the channel access mechanism of the protocol.

The results about the packet delivery delay in Fig. 20 also support these facts. The picture suggests that the delay for delivering 100-byte packets is expectedly the lowest, whereas the delay affecting the delivery of 1000-byte packets is the highest. Nevertheless, we observe that the latter is about 25 times the former, i.e., the delivery delay is not directly proportional to the ratio of the packet sizes. This increase is due to the fact that a packet transmission, in the 1000-byte packet case, is subject to further delays caused by repeated channel sensing until the channel is found free.
We now discuss the energy consumption when using the CSMA protocol without error control. Figs. 21, 22 and 23 show the energy consumption per node for transmissions, receptions and idling, respectively. We recall from Section 5 that the simulation time is set so that each node generates on average 100 packets. For this reason, the transmit and receive energy consumption are constant for all packet sizes. In particular, the latter is expectedly higher than the former, as a packet is transmitted by a single node, but is received by multiple neighbors. For the same reason, we observe that the energy spent in idling is inversely proportional to the value of the packet generation rate $\lambda$, as the duration of the simulation is higher when $\lambda$ is small, in order to allow all nodes to send on average 100 packets. From the analysis of the energy consumption made this way, we can infer that the energy consumption due to transmissions and receptions is one order of magnitude lower than the energy consumed when the node is idle. In turn, the idling energy consumption is one order of magnitude lower than the capacity of the battery. Therefore, assuming that the energy available to the modem is actually one half of the nominal battery capacity, we can predict that a cNode would survive MAC trials for about 60 days in the worst case, i.e., at the highest level of traffic ($\lambda = 0.2$ packets/minute/node, or one packet every five minutes per node) and for the highest packet size (1000 bytes). This figure is extrapolated from the total energy consumption computed at $\lambda = 0.2$, which amounts approximately to 12 kJ (of which 3 kJ for transmissions, 4 kJ for receptions and 5 kJ for idling). Given a battery capacity of 2 MJ, this tells us that the simulation could last $2 \cdot 10^6 / 12 \cdot 10^3 \approx 167$ times the duration of the simulation for $\lambda = 0.2$. We recall that the latter is chosen so that a node generates on average 100 packets, at a pace of one packet every 5 minutes (or 300 seconds) plus 1000 seconds to serve the packets remaining in the queue of the nodes. The total of the above is 31000 seconds, or 517 minutes. Hence, $517 \times 167 \approx 86340$ minutes, or about 60 days.

The last result that we show in Fig. 24 is the ratio between the packets dropped for harmful interferences in the network, and the number of total packets dropped (for noise, interference, deafness of the receiver). First of all, we can see how the ratio of packet dropped for interference is increasing as the traffic, $\lambda$, increase (as one would expect). However, even at the higher value of the traffic considered, the packets dropped due to interference is less than the half of all the packets dropped. This means that the short carrier sense provided by CSMA is very effective in preventing most of the harmful collisions that may occur, because more than half of the packets dropped is due to causes not attributable to the MAC protocol (e.g., for noise in the channel or for deafness of the receiver because it is busy for another reception or transmission process). In particular, we can see that, for packets of 500 and 1000 bytes, the percentage of packets dropped due to interference is about 35%, while, for packets of 100 bytes, is slightly higher (about 45%), because shorter packets make the CSMA mechanism less capable to detect possible interferers.
Fig. 18: Throughput per node as a function of the packet generation rate per node for CSMA without error control.

Fig. 19: Packet delivery ratio as a function of the packet generation rate per node for CSMA without error control.

Fig. 20: Packet delivery delay as a function of the packet generation rate per node for CSMA without error control.
Fig. 21: Energy spent for transmissions as a function of the packet generation rate per node for CSMA without error control.

Fig. 22: Energy spent for receptions as a function of the packet generation rate per node for CSMA without error control.

Fig. 23: Energy spent for idling as a function of the packet generation rate per node for CSMA without error control.
Fig. 24: Ratio between the number of total packets dropped and the number of packet dropped for interference for CSMA without error control.
7.2 CSMA with S&W ARQ

We consider now Figs. 25 to 31, which report the results of the CSMA MAC protocol with a Stop-and-Wait ARQ policy. For each packet, 5 transmission attempts per packet are performed overall (i.e., the first transmission plus up to 4 retransmissions).

The behavior of the throughput curves in Fig. 25 is analogous to the CSMA version without error control in Fig. 18: the only visible difference is in the throughput for the 1000-byte packet case at $\lambda = 0.2$. Due to the presence of retransmissions, for a given value of $\lambda$, a higher quantity of traffic injected in the network with respect to the case with no error control. As expected, retransmissions improve the the probability of delivering a packet correctly, at least for sufficiently small values of $\lambda$. This can be seen in Fig. 26, where the PDR is shown to be initially very high, and to decrease only at high traffic. We notice that, even in the presence of retransmissions, the PDR at the highest values of $\lambda$ is lower than in the case without error control. This result is due to the ACK packets required to confirm correct reception. While on one hand they are beneficial if the traffic is not too intense, on the other hand they may make the sink temporarily deaf to packet transmissions. In any event, the PDR never decreases below 0.78, even for the largest packet size. Fig. 27 shows that, on average, 2 to 3 retransmissions are required to transmit a packet correctly. We recall that the maximum number of retransmissions is 4. While it may seem counter-intuitive that this number is not reached in Fig. 27, we note that packets are lost not only when the maximum number of retransmissions is reached: for example, a newly generated packet may be dropped because the buffer of the generating node is full. This would count in the packet generation rate, but not in the average number of retransmissions.

In the presence of retransmissions, a larger packet delivery delay can be expected. This is observed in Fig. 28, especially for the case of 1000-byte packets. On average, these delays are larger than the delivery delay of the case without error control (see Fig. 20) times the number of retransmissions, because of the delays introduced by backoff procedures and further channel access attempts.

The energy consumption per node is reported in Figs. 29, 30 and 31 (showing the transmit, receive and idle energy consumption, respectively). The same observations made for the case of CSMA without error control still apply here. However, we note that the transmit and receive energy consumption are higher (due to retransmitted data packets). Correspondingly, this causes the energy consumption to be dominated by transmissions and receptions for $\lambda = 0.2$. The expected worst-case node lifetime (computed analogously to the case without error control in the previous section) is approximately 36 days, assuming a worst-case energy consumption of 10 kJ for transmissions, 15 kJ for receptions, and 5 kJ for idling.

The ratio between the packets dropped for interference and the total number of packets dropped, Fig. 32, is, even in this case, less than 0.5. We can notice some differences between this case and the previous case without error control (Fig. 24). In particular, we can see that the ratio increases very quickly as the traffic $\lambda$ increases, but saturates at about $\lambda = 0.06$ packets/minute/node. This quick increase of
the ratio of packets dropped for interference is due to re-transmissions required by the S&W ARQ scheme implemented over CSMA protocol. In fact, as the traffic increases, a collision is more likely; hence, a re-transmission required by the protocol is also more likely. In this manner, the traffic in the network increases rapidly and the number of collisions increases accordingly. Nevertheless, this process reaches a sort of stability, and the ratio remains steady for each value of traffic considered. The maximum value reached is approximately similar to the one in the previous case without error control.
Fig. 25: Throughput per node as a function of the packet generation rate per node for CSMA with error control.

Fig. 26: Packet delivery ratio as a function of the packet generation rate per node for CSMA with error control.

Fig. 27: Number of link-level retransmissions per packet as a function of the packet generation rate per node for CSMA with error control.
Fig. 28: Packet delivery delay as a function of the packet generation rate per node for CSMA with error control.

Fig. 29: Energy spent for transmissions as a function of the packet generation rate per node for CSMA with error control.

Fig. 30: Energy spent for receptions as a function of the packet generation rate per node for CSMA with error control.
Fig. 31: Energy spent for idling as a function of the packet generation rate per node for CSMA with error control.

Fig. 32: Ratio between the number of total packets dropped and the number of packet dropped for interference for CSMA with error control.
7.3 DACAP without error control

The results regarding the DACAP protocol without error control are reported in Figs. 33 to 38. We recall that DACAP, in this case, is based on a three-way handshaking procedure that requires the transmission of an RTS and a CTS before a data packet is sent. The presence of a handshake requires that a full round-trip time (RTT) elapses from the beginning of channel access procedures to when a data packet can actually be transmitted. This turns into a throughput that is lower than CSMA’s throughput. The difference is minimal because the network is deployed over a limited area, hence the RTTs are low with respect to the packet transmission time considered in the simulations.

What changes considerably with respect to CSMA is the PDR (Fig. 34), which is sensibly lower, and the delay per km (Fig. 35), which is much higher than what seen for CSMA in Fig. 20. The main reason behind this behavior is the increased level of contention. Where CSMA transmitted no signaling packets and could suffer only from interference among concurrent data packet transmissions, DACAP’s communications may disrupt one another in several ways. For example, a collision between signaling packets, as well as between signaling and DATA packets, may be just as dangerous as a collision between data packets (which is less likely here due to preliminary handshakes). Errors generate backoffs, which, when repeated, tend to increase the waiting time. In addition, we note that the delay curves in Fig. 35 are inversely proportional to $\lambda$. This is the same behavior observed for the idling energy consumption. In fact, we recall that MAC-level queues do not expire, i.e., a packet may stay in the buffer of the MAC protocol until it is transmitted: due to high contention, with DACAP some nodes may seize the channel for a long time. Hence, the event that some nodes are incapable of transmitting and hold their packets in queue for a long time is quite frequent with DACAP. This event is the reason behind the behavior of the delay curves. The same event also explains why simulations with 100-byte packets yield longer delays: as the MAC buffer size is fixed, the number of packets with 100-byte packets that can be stored therein is 10 times as much as the number of packets with 1000-byte packets. In turn, this causes the delay to increase considerably.

The fact that some nodes are forced to prolonged and repeated backoffs is also confirmed by looking at the transmit and receive energy consumption in Figs. 36 and 36 which tend to decrease for increasing $\lambda$, unlike CSMA’s which stayed constant. The behavior of the idling energy consumption is instead the same as in Fig. 37. The expected worst-case node lifetime when DACAP with no error control is used is hence about 60 days (assuming a worst-case energy consumption of about 3 kJ for transmissions, 4 kJ for receptions and 5 kJ for idling).

The ratio between the packets dropped for interference and the total number of packets dropped, Fig. 39, is quite steady for every traffic value, $\lambda$. Moreover, we can see that, even for very low traffic value, the ratio is higher than 0.5. In particular, for the case of packets of 500 and 1000 bytes, almost all the packets dropped in the simulation are dropped because of harmful collisions. This means that DACAP cannot manage the increasing traffic and, moreover, even at very low traffic loads...
in the network, it cannot assure a safe access to the channel, especially in dense networks like the scenario considered in our case. This behavior is due to the well known hidden terminal problem and the large propagation delays that makes the RTS/CTS method ineffective in underwater scenarios.

Moreover, even short control messages like RTS and CTS can be harmful for a data packet. In particular, we can see that, unlike in the CSMA case (Figs. 24 and 32) using larger data packets (500 or 1000 bytes) brings to higher percentage of packets dropped for collisions (slightly less than 85% versus about 60%). A packet of 1000 bytes takes longer to be transmitted, hence it is more likely that another node attempts to access the channel, causing harmful interference, even if the control packet is usually smaller than the data packet.
Fig. 33: Throughput per node as a function of the packet generation rate per node for DACAP without error control.

Fig. 34: Packet delivery ratio as a function of the packet generation rate per node for DACAP without error control.

Fig. 35: Packet delivery delay as a function of the packet generation rate per node for DACAP without error control.
Fig. 36: Energy spent for transmissions as a function of the packet generation rate per node for DACAP without error control.

Fig. 37: Energy spent for receptions as a function of the packet generation rate per node for DACAP without error control.

Fig. 38: Energy spent for idling as a function of the packet generation rate per node for DACAP without error control.
Fig. 39: Ratio between the number of total packets dropped and the number of packet dropped for interference for DACAP without error control.
7.4 DACAP with S&W ARQ

Figs. 40 to 46 show the performance of DACAP with stop-and-wait ARQ for error control. Most considerations already made when comparing the two versions of CSMA with and without error control also apply to the comparison with their DACAP counterparts. In particular, we observe that retransmissions help DACAP achieve a quite better PDR than without error control. This is confirmed by the PDR curves reported in Fig. 41.

Fig. 43 expectedly shows higher delays than in Fig. 35 due to the presence of retransmissions. The observation that for increasing $\lambda$ some nodes tend to seize the channel also applies here, and is confirmed by the fact that the delay is roughly inversely proportional to $\lambda$. This is in line with the case of DACAP without error control. We also note that the number of retransmissions (Fig. 42) is less than the maximum amount, even in the presence of a PDR lower than 1. This is due to the fact that a packet is discarded after a maximum number of attempts has been reached. This number includes both retransmissions (i.e., successful channel access and transmissions for which no ACK is received) as well as unsuccessful channel access attempts (for which no packet is transmitted).

Figs. 44 and 45 show that the energy consumption is still roughly constant as for CSMA. We note, however, that the heavier traffic induced by the 1000-byte packets, in addition to the RTS/CTS-based channel access, still leads the network closer to congestion, as seen from the fact that the number of transmissions (hence the transmit energy consumption) decreases for increasing $\lambda$ due to a higher number of backoff events. The expected worst-case node lifetime for DACAP with S&W ARQ is approximately 51 days (assuming a worst-case energy consumption of about 3.5 kJ for transmissions, 5.5 kJ for receptions and 5 kJ for idling).

For what concerns the ratio of packets dropped for interference (Fig. 47), we can make the same considerations of the previous case (Fig. 39). However, there are some differences. First of all, in this case, the ratio of packet dropped for interference is increasing as the traffic load increases, because the S&W ARQ scheme implemented causes re-transmissions that contribute to increasing the traffic in the network and the probability of a collision (like in the case of CSMA with error control). However, the higher value is roughly similar to the case without ARQ.
Fig. 40: Throughput per node as a function of the packet generation rate per node for DACAP with error control.

Fig. 41: Packet delivery ratio as a function of the packet generation rate per node for DACAP with error control.

Fig. 42: Number of link-level retransmissions per packet as a function of the packet generation rate per node for DACAP with error control.
Fig. 43: Packet delivery delay as a function of the packet generation rate per node for DACAP with error control.

Fig. 44: Energy spent for transmissions as a function of the packet generation rate per node for DACAP with error control.

Fig. 45: Energy spent for receptions as a function of the packet generation rate per node for DACAP with error control.
Fig. 46: Energy spent for idling as a function of the packet generation rate per node for DACAP with error control.

Fig. 47: Ratio between the number of total packets dropped and the number of packet dropped for interference for DACAP with error control.
7.5 Tone-Lohi without error control

Unlike DACAP’s channel access procedure, the method employed in Tone-Lohi involves only tones (not packets) and is driven only by transmitters. Receivers take no part in the contention process. This typically helps speed up channel access, especially in networks deployed over small areas and where the main communication pattern is many-to-sink and, in fact, in this case the transmitters can likely hear one another’s tone. This immediately yields a better throughput than with DACAP’s 3- or 4-way handshaking, as seen in Fig. 48, where the throughput is comparable to that of CSMA and the linear behavior of the curves for all packet sizes confirms that the network is not congested. The same conclusion can be drawn from the PDR curves in Fig. 49, where the curves are almost constant and mildly decrease for increasing \( \lambda \) only for packet sizes of 500 and 1000 bytes. Notably, Tone-Lohi’s tone transmission help avoid some collision events, hence the PDR for packet sizes of 1000 bytes is better with Tone-Lohi than with CSMA.

The behavior of the packet delivery delay in Fig. 50 is also comparable to that of CSMA (see Fig. 20, with the exception that the tone transmission and the subsequent contention resolution procedures take time, hence the delays of Tone-Lohi are slightly higher than CSMA’s. On the contrary, the energy consumption in transmission, reception, and idling (Figs. 51, 52 and 53, respectively) are entirely analogous to those of CSMA, meaning that the two protocols require almost the same number of transmissions.

While Tone-Lohi’s procedure is certainly lighter than DACAP’s, receivers never take part in the channel access procedure. Therefore, Tone-Lohi may be more prone to hidden terminal effects, and yield worse multihop performance in turn. This intuition is confirmed by several results reported in the literature, such as those reported in [27, 39] support this statement. Nevertheless, Tone-Lohi remains a good candidate for data retrieval networks deployed over small areas, where the communication pattern is mostly many-to-sink.

Analogously to CSMA without error control, the expected worst-case node lifetime is approximately 60 days (assuming a worst-case energy consumption of about 3 kJ for transmissions, 4 kJ for receptions and 5 kJ for idling).

For what concerns the ratio of packet dropped due to interference, Tone-Lohi, in this case, exhibit the lowest ratio of packet dropped for interference, if compared to CSMA and DACAP, even if CSMA exhibits better throughput due to the fact that it bears no overhead in terms of controls packet required for access the channel and hence has the lower channel access latency. In particular, the ratio of packet dropped due to interference is less than 10% in the case of 100 byte payloads, whereas is is at most 20% for the higher values of \( \lambda \) in the case of 500- and 1000-byte payloads. This shows the effectiveness of Tone-Lohi in preventing collisions. In particular, the light contention driven by the transmitter makes it possible to avoid most of the collisions that may occur in the network. Moreover, increasing the traffic does not bring to a very quick increase of the percentage of packets dropped for interference.
Fig. 48: Throughput per node as a function of the packet generation rate per node for Tone-Lohi without error control.

Fig. 49: Packet delivery ratio as a function of the packet generation rate per node for Tone-Lohi without error control.

Fig. 50: Packet delivery delay as a function of the packet generation rate per node for Tone-Lohi without error control.
Fig. 51: Energy spent for transmissions as a function of the packet generation rate per node for Tone-Lohi without error control.

Fig. 52: Energy spent for receptions as a function of the packet generation rate per node for Tone-Lohi without error control.

Fig. 53: Energy spent for idling as a function of the packet generation rate per node for Tone-Lohi without error control.
Fig. 54: Ratio between the number of total packets dropped and the number of packet dropped for interference for Tone-Lohi without error control.
7.6 Tone-Lohi with S&W ARQ

Figs. 55 to 61 show the performance of Tone-Lohi with S&W ARQ. Analogous to CSMA, Tone-Lohi shows slightly lower throughput than its version without error control, mainly due to the further delay required to wait for an ACK from the sink. As for CSMA, ACKs help achieve better transmission performance to some extent but, in the presence of heavier traffic, they also tend to be a source of collisions. This can be seen in Fig. 56, where the PDR is very close to 1 for all values of $\lambda$ for packet sizes of 100 and 500 bytes, but much lower for 1000-byte packets in the presence of heavy traffic, e.g., $\lambda = 0.2$. In fact, the combined effect of tone transmissions (hence fewer collisions), short packets (hence lower chances of errors due to noise) and of S&W ARQ (hence correction of the few errors that still take place) makes channel access very effective for 100-byte and 500-byte packets. For 1000-byte packets, instead, the presence of ACKs becomes a source of further collisions and loss of packets, even worse than for CSMA with S&W (Fig. 56).

The delay observed in Fig. 58 is also analogous to that of CSMA (Fig. 28) and the same considerations apply. The curve for the 1000-byte packet case tends to increase with increasing $\lambda$ because of the higher amount of time passed in queue by a packet, on average. We also observe that the number of retransmissions per packet (Fig. 57) does not reach the maximum possible value in spite of a PDR which may be lower than 1. In fact, a packet is discarded when an overall maximum number of attempts is reached, which includes retransmissions in case of missing ACKs, but also failed channel access attempts. This is in line with the conclusions drawn for DACAP.

Again, we observe that the network easily gets congested in the 1000-byte case, mainly because of the heavy contention for channel access. This can also be seen in Figs. 59 and 60, showing that the transmit and receive energy consumption, respectively, decrease with increasing $\lambda$ for the 1000-byte packet case (due to the reduced number of transmissions performed).

The expected worst-case node lifetime in this case is approximately 32 days (assuming a worst-case energy consumption of about 3 kJ for transmissions, 4 kJ for receptions and 5 kJ for idling).

As regards the ratio of packets dropped because of collisions, we can make the same consideration that we made for Tone-Lohi without error control. The cases with payloads of 500 and 1000 bytes exhibits a quick increase as the traffic increases, for the reasons already explained for the case of CSMA and DACAP with error control. The value of the ratio between the packets dropped for collisions and the total number of collision is higher than for CSMA and DACAP (almost double). Anyways, the percentage is less than 50%.
Fig. 55: Throughput per node as a function of the packet generation rate per node for Tone-Lohi with error control.

Fig. 56: Packet delivery ratio as a function of the packet generation rate per node for Tone-Lohi with error control.

Fig. 57: Number of link-level retransmissions per packet as a function of the packet generation rate per node for Tone-Lohi with error control.
Fig. 58: Packet delivery delay as a function of the packet generation rate per node for Tone-Lohi with error control.

Fig. 59: Energy spent for transmissions as a function of the packet generation rate per node for Tone-Lohi with error control.

Fig. 60: Energy spent for receptions as a function of the packet generation rate per node for Tone-Lohi with error control.
Fig. 61: Energy spent for idling as a function of the packet generation rate per node for Tone-Lohi with error control.

Fig. 62: Ratio between the number of total packets dropped and the number of packet dropped for interference for Tone-Lohi with error control.
Fig. 63: A scheme of the node deployment in the new scenario derived from Scenario 1. Every box contains the ID of the node randomly placed within the box. The sink is always placed in box 5, 2000 m north and 2000 m west of the bottom-left corner of the picture, 10 m below the surface, and is represented here as a surface buoy.

7.7 On the effect of S&W ARQ on larger scenario

In this section, we enlarge the 10 nodes scenario single-hop from a square of 1 km × 1 km (Fig. 7) to a square of 4 km × 4 km (Fig. 63) to highlight the effects of the S&W error control implemented over the MAC protocols considered, with larger propagation delays and larger distances among nodes.

Like in the previous scenario described in Section 4, the area is divided in 9 boxes with equal size. Every box contains a node randomly placed into the box. The sink is placed in the center of the area, 10 m below the surface. The parameters used are the same of the previous scenario but the transmission power, that is adapted so that every node can reach the sink. We calculated a suitable transmission power for the new distances between the nodes and the sink using the formula

$$P_{\text{tx}} = P'_{\text{tx}} \cdot \left(\frac{d'}{d}\right)^k \cdot a(f)^{d'-d}$$

(6)

where $P'_{\text{tx}}$ is the previous transmission power used (set to 178 dB re $\mu$Pa), $d'$ is the previous mean distance between the nodes and the sink and $d$ is the actual mean distance between the nodes and the AUV. $a(f)^8$, instead, is the Thorp’s formula which gives us the attenuation in function of the frequency used. The path-loss exponent, $k$, used is 1.865. The procedure to retrieve this value is explained better in the following subsection of UW-Polling simulations. Replacing in the formula the values for our scenario, it turns out that the ratio between the old and the new transmission power should be

$$\frac{P_{\text{tx}}}{P'_{\text{tx}}} = \left(\frac{2}{0.5}\right)^{1.865} \cdot a(f)^{(2-0.5)} \approx 120$$

(7)

$^8$ The Thorp’s formula is $10 \cdot \log(a(f)) = 0.11 \cdot \frac{f^2}{1 + f^2} + 44 \cdot \frac{f^2}{4100 + f^2} + 2.75 \cdot 10^{-4} f^2 + 0.03$
translating this value into dB scale

\[ 10 \cdot 10^{\log(120)} \approx 21 \text{ dB} \] (8)

Following the previous calculations, we set the new transmission power to 200 dB re \( \mu \text{Pa} \). The others simulations parameters are left as in the previous case but the Transmit power, which is adapted for the new SPL (which the maximum SPL that the KM modem can use) and is 150 W. Here we report, for completeness, the whole list of simulations parameters with the new transmission power and the new transmit power:

- **Bit rate**: 1000 bps
- **Carrier frequency**: 25.6 kHz
- **Bandwidth**: 4 kHz
- **Source Power Level (SPL) at 1 m**: 200 dB re \( \mu \text{Pa} \)
- **Transmit / Receive / Idle power**: 150 W, 620 mW, 85 mW
- **No sleep mode**
- **Modulation**: BPSK
- **Acquisition threshold**: 1 dB
- **Node battery capacity**: 78 Ah; at an output voltage of 14.4 V, this translates into a nominal maximum of about 4 MJ of available energy.\(^9\)

The metrics considered for evaluate the protocols performance are the same of the previous case. In the following sections, we present the plots for the all the protocols considered (CSMA, DACAP and Tone-Lohi).

\(^9\) KM states that this battery capacity is a conservative estimate, and can hence be used directly. In the following, we will assume that only half of this capacity (2 MJ) is available to the modem, while the rest is required to operate other components of the CLAM node, including the sensors and the embedded computer controlling the modem.
We consider CSMA protocol with S&W ARQ policy in the new scenario presented above. As we can see in Fig. 64, the throughput increases as it happens for the previous scenario (Fig. 25), but is lower for the case with of 1000-byte payloads. Taking a look at the Packet Delivery Ratio (Fig. 65), we can see that the maximum value of PDR is 0.95 (while in the previous case was 1). This means that, also at the lower value of traffic, \( \lambda \), not all the packets can be delivered correctly to the destination (and only few packets are dropped for interference as we can notice on Fig. 71); in this scenario, where the distance between nodes and the sink is significantly increased, even at low traffic some packets may be lost due to the attenuation of the channel or the noise. Nevertheless, the ARQ scheme implemented over the MAC protocol, makes it possible to achieve a good packet delivery ratio anyways. In fact, even at high traffic, where also the interference in the network plays a not negligible role, the PDR goes slightly under 0.65, thanks to the error control that increases the probability to deliver correctly the packet. We can also observe that, with a payload of 1000 byte, with high traffic load, we need on average 5 retransmissions for correctly delivering the packets. This proves the fact that, without acknowledgements, in this scenario, the Packet Delivery Ratio would be lower. The ratio of packet dropped for interference (Fig. 71) is, also in this case, lower than 0.5 and is lower than in the case with the 1 km \times 1 km area because the nodes farther apart, so the interference generated by the transmission of other nodes may be less harmful. Moreover, also in this case CSMA can deal well with the interference generated by the network itself.

Taking a look to the transmit and receive power (Figs. 68 and 69 respectively), we note that the nodes spend significantly more energy both in reception and transmission. This is because the nodes consume much more power in the transmission phase because of the new SPL adopted to cope with the increased distance. The new energy consumption in the reception phase is due to the large number of retransmissions required (at least 3 on average for the best case). This means that the lifetime of the battery of the nodes in this scenario would be significantly lower, as one would expect. The end-to-end delivery delay (Fig. 67), instead, is slightly lower than the previous case (Fig. 28), but, the more apparent difference is the trend of the delay as the traffic increases. While in Fig. 28, the delay increases slowly as the traffic increases, in this case the delay reaches a maximum (slightly lower than the maximum in the previous case) and then falls to values that are around 10 seconds, except for the case with payloads of 1000 bytes, in which the delay increases slowly. This behavior of the delay with payloads of 1000 byte can be explained remembering that the sender needs a large number of re-transmission for delivering the packet correctly, hence the delay between the first transmission and the final (hopefully) correct reception, increases. In the other cases (especially for the case with payload of 100 byte), the end-to-end delivery delay remains constant over all the traffic load considered. The increased size of the network decreases the average amount of interference affecting the transmissions, so that is more likely that the nodes sense the channel free and so the node has to wait less time to transmit the packet.
Fig. 64: Throughput per node as a function of the packet generation rate per node for CSMA with error control.

Fig. 65: Packet delivery ratio as a function of the packet generation rate per node for CSMA with error control.
Fig. 66: Number of link-level retransmissions per packet as a function of the packet generation rate per node for CSMA with error control.

Fig. 67: Packet delivery delay as a function of the packet generation rate per node for CSMA with error control.
Packet generation rate per node, $\lambda$ [packets/minute]

Energy Consumption per node (Transmission only) [J]

<table>
<thead>
<tr>
<th>Payload</th>
<th>100 byte</th>
<th>500 byte</th>
<th>1000 byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 68: Energy spent for transmissions as a function of the packet generation rate per node for CSMA with error control.

Packet generation rate per node, $\lambda$ [packets/minute]

Energy Consumption per node (Reception only) [J]

<table>
<thead>
<tr>
<th>Payload</th>
<th>100 byte</th>
<th>500 byte</th>
<th>1000 byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 69: Energy spent for receptions as a function of the packet generation rate per node for CSMA with error control.
Fig. 70: Energy spent for idling as a function of the packet generation rate per node for CSMA with error control.

Fig. 71: Ratio between the number of total packets dropped and the number of packet dropped for interference for CSMA with error control.
7.7.2 DACAP with S&W ARQ

Most of the considerations made for the CSMA case can be made for DACAP as well. In particular, with respect to the scenario 1 km × 1 km, we observe a worse throughput and Packet Delivery Ratio than for CSMA. In this case, like in the previous case when the size of the network was smaller, DACAP does not reach the maximum number of re-transmission and, in most cases, the number of retransmission is equal to zero: this is because the packet is discarded after a maximum number of channel access attempts has been reached. If we focus on the ratio of packet dropped for interference (Fig. 79), we note that most part (in some cases the whole part) of the packets is dropped because of harmful interference, even if the network deployment is larger. This result show us that a large amount of interference is present in the network, also because of the various channel attempts made by the node and several control packets such as RTS and CTS are traveling around the network. This explains better how, even if the Packet Delivery Ratio is lower than 0.5 for several values of the traffic load, the number of link-level re-transmissions is low.

The end-to-end delay, Fig 75, has the same shape of CSMA because, as already said, some nodes tends to seize the channel access. Notably, the delay is significantly lower than in the case of the smaller networks. This result seems to be counter-intuitive. However, we have to recall that the end-to-end delivery delay is calculated only for the packets that are correctly delivered to the sink, which are only a very small fraction of the total. For energy consumption we can see at Figures 76, 77 and 78 for energy consumption during transmission, reception and idling, respectively. The nodes spend more energy than in the case with smaller network (Fig. 44) because the nodes both transmit at higher power (as we explained above for CSMA) and also because of the large amount of channel attempts and hence the large number of control packets that are transmitted. On the contrary, the energy spent for reception is lower than in the previous case (see Fig. 45). This result is explicable by recalling that the Packet Delivery Ratio in this case is lower, hence, fewer data packets are correctly delivered to the destination. For this reason, on average, the nodes spend less time in the reception phase. Also in this case the energy consumption both in reception and transmission phases is roughly constant over all the traffic loads considered.
Fig. 72: Throughput per node as a function of the packet generation rate per node for DACAP with error control.

Fig. 73: Packet delivery ratio as a function of the packet generation rate per node for DACAP with error control.
Fig. 74: Number of link-level retransmissions per packet as a function of the packet generation rate per node for DACAP with error control.

Fig. 75: Packet delivery delay as a function of the packet generation rate per node for DACAP with error control.
Fig. 76: Energy spent for transmissions as a function of the packet generation rate per node for DACAP with error control.

Fig. 77: Energy spent for receptions as a function of the packet generation rate per node for DACAP with error control.
Fig. 78: Energy spent for idling as a function of the packet generation rate per node for DACAP with error control.

Fig. 79: Ratio between the number of total packets dropped and the number of packet dropped for interference for DACAP with error control.
7.7.3 Tone-Lohi with S&W ARQ

Figs. 80 and 81 show the throughput and the Packet Delivery Ratio of Tone-Lohi protocol. We can draw the same considerations made with CSMA and DACAP in this case with respect to their counterpart with the larger network. Furthermore, we can note that the channel access provided by Tone-Lohi is effective, at least for 100-byte and 500-byte payloads cases. The 1000-byte case exhibits a lower Packet Delivery Ratio, which is similar to the one of CSMA. The packet delivery delay depicted on Fig. 83 is the lowest if compared with the end-to-end delivery delay of CSMA (Fig. 67) and DACAP (Fig. 75). The delay is roughly constant for the case of 500- and 100-byte payloads, whereas it increases for increasing λ for the 1000-byte case, where the number of contentions for channel access is higher. The main difference that we can note from the case with smaller network is the number of data-link retransmissions. In this case we have a quite large amount of re-transmissions, if compared with the case with smaller network (Fig. 58. This behavior is due to the fact that more packets than in the case of smaller network are lost due to noise or interference (even if less than half of the amount of erroneous packets are lost for interference, as we can see on Fig. 87) and, hence, more re-transmissions are required by Stop-And-Wait paradigm to deliver correctly the packet. The energy consumption during transmission, reception and idling are depicted on Figs. 84, 85 and 86. We can note that the energy consumption in transmission phase is greater if compared with the case with smaller network (Fig. 59) for the same reasons explained above, while the energy spent for reception is the same (Fig. 60). Notably, also in this case we have that the energy consumption is more quickly increasing than the other two cases as traffic increase because of the larger number of re-transmission required for correctly delivering a packet.
Fig. 80: Throughput per node as a function of the packet generation rate per node for Tone-Lohi with error control.

Fig. 81: Packet delivery ratio as a function of the packet generation rate per node for Tone-Lohi with error control.
Fig. 82: Number of link-level retransmissions per packet as a function of the packet generation rate per node for Tone-Lohi with error control.

Fig. 83: Packet delivery delay as a function of the packet generation rate per node for Tone-Lohi with error control.
Fig. 84: Energy spent for transmissions as a function of the packet generation rate per node for Tone-Lohi with error control.

Fig. 85: Energy spent for receptions as a function of the packet generation rate per node for Tone-Lohi with error control.
Fig. 86: Energy spent for idling as a function of the packet generation rate per node for Tone-Lohi with error control.

Fig. 87: Ratio between the number of total packets dropped and the number of packet dropped for interference for Tone-Lohi with error control.
7.8 Concluding remarks on the simulations of CSMA, DACAP and Tone-Lohi

In this section we have simulated the performance of three MAC protocols for underwater acoustic networks, CSMA, DACAP and Tone-Lohi, in scenarios of interest for the CLAM project. These protocols have been chosen because they prescribe different levels of channel access (channel sensing, RTS/CTS-based handshaking and transmitter-side tone-based contention, respectively), and thus allow to compare different approaches to underwater channel access. The scenarios and modem parameters have been agreed on in collaboration with WP2 and WP3. The results suggest that CSMA provides the best overall performance, and thus constitutes the best candidate for use in conjunction with routing protocols. This is the reason why, e.g., the SUN routing protocol presented in Section 3.1 is built on top of CSMA. Tone-Lohi is the most direct competitor of CSMA in our comparison. The results suggest that, especially with reference to the S&W ARQ versions, its packet delivery ratio is slightly better and that, for 1000-byte packets, the delay experienced by the packets is lower than for CSMA. However, prompted by other contributions such as [39,40], we prefer not to employ Tone-Lohi in multihop scenarios because of its vulnerability to hidden terminal problems. This decision also comes from the following practical issue: Tone-Lohi is completely effective only if contentions are driven by tones. This usually requires a separate transceiver system. However, the KM modem available in CLAM has only one transceiver, with its own physical layer algorithms. This transceiver cannot be reconfigured to transmit and detect tones. Tone-Lohi could still be run on KM’s modem if tones are implemented as short packet transmissions, as in [41]. However, this makes the performance of Tone-Lohi more akin to that of DACAP, since these “packet” tones can now create substantial interference to data packets. For the reasons above, CSMA (with ACKs) is a better choice for medium access control procedures.

7.9 UW-Polling simulations

7.9.1 Introduction

In this section, we detail the results of UW-Polling simulations. Given the target use cases of UW-Polling explained at the beginning of this section, we will simulate UW-Polling in scenarios 2 and 3 instead of scenario 1. In fact, UW-Polling is a MAC protocol, but finds its best application in data retrieval scenarios where the network is not capable of multihop communications (e.g., because it is not fully connected) or when the loss of nodes has to be compensated for. In addition, one of the key features of UW-Polling is that the AUV can manage the time-varying topology it perceives as it moves along its trajectory (because some nodes would enter and some would exit the AUV coverage range as it moves): if we simulated
UW-Polling in scenario 1, these topology changes would have been minimal. In scenarios 2 and 3, instead, the network is deployed over a large area and is not fully connected (unlike in scenario 1), hence the simulations results are more significant.

### 7.9.2 A note on the simulations

Before moving on to the plots of the network metrics, we would like to stress that the simulation of a mobile network is no short task with nsMiracle [16] and WOSS [1]. We recall here that nsMiracle is the network simulator of choice in CLAM and WOSS is its interface to the Bellhop ray tracing tool [17]. In particular, Bellhop is employed here in order to retrieve channel power gains and thereby estimate the signal-to-noise-and-interference ratio at the receiver, from which the probability that a wanted packet is received correctly can be inferred.

When the AUV moves, the environment around it (e.g., the local bathymetry of the surrounding area) changes. For this reason, a fresh Bellhop run has to be made periodically, in order to re-compute the channel power gain between the AUV and the static network nodes. Every Bellhop run takes from several seconds to several minutes to complete, depending on the complexity of the environment, on the precision of the environmental parameters (e.g., the number of samples of the sound speed profile along the water column), on the distance between the nodes, on the number of rays to be traced in order to achieve a sufficient accuracy. In addition, the number of Bellhop runs increases linearly with the number of nodes in the network: in fact, a separate Bellhop run has to be performed for the wanted receiver as well as for every other node, in order to compute the amount of interference caused by the ongoing transmission.

One way to reduce the complexity of the problem is already implemented in WOSS, and involves forcing a given “spatial coherence,” i.e., avoiding to re-run Bellhop every time the transmitter is found within a given distance of any position from where a Bellhop run has already been calculated. While this approach reduces the number of Bellhop runs, full-scale network simulations may still take several days to complete.

In WP3, we showed that it is possible to choose the path-loss exponent $b$ of the Urick empirical attenuation model [15, 22], in order to approximate the attenuation yielded by Bellhop for several values of the source depth, of the destination depth, and of the distance between the source and the destination. For completeness, the model we are referring to is

$$A(R, f) = R^b a(f)^R,$$  \hspace{1cm} (9)

where $R$ is the distance between the source and the destination in meters, $b$ is the path-loss exponent, $f$ is the carrier frequency, and $a(f)$ is the frequency-dependent absorption loss expressed in m$^{-1}$. The model for $a(f)$ employed in Bellhop and also considered for this analysis is
\[ a_{\text{dB}}(f) = \left( \frac{40f^2}{4100 + f^2} + 0.1f^2 \right) \cdot \frac{1}{914.4}, \]  

where 914.4 is the ratio of 1000 yards to 1 m, and \( a_{\text{dB}}(f) = 10 \log_{10} a(f) \). Simply setting the path-loss exponent of the model to an appropriate value allows the channel power gain predicted by the empirical model and by bellhop to be similar, yielding in turn matching network simulation results. The whole discussion will be reported in D3.2, and is part of a journal paper submitted to the IEEE Transactions on Wireless Communications, namely “The Throughput of Underwater Networks: Analysis and Validation using a Ray Tracing Simulator” (see also D6.3 and the M14–M20 progress report). For the present discussion, let us outline the procedure to find the optimal path-loss exponent.

To fix ideas, go back to Fig. 12, where the white spot at the center of the picture denotes the location of the nodes considered in the simulations. The west-east and south-north bathymetry profiles crossing at that point are found in Fig. 13 and 14, respectively. In these pictures, the network of scenarios 2 and 3 is deployed around the flat area in Fig. 14, and around the range from 278 to 282 km in Fig. 13, where the bathymetry mildly increases towards the east. Focus for the moment on the west-east bathymetry, and assume that the AUV is moving towards east at a constant depth of 50 m.

While the depth of the transmitter is fixed, the sensor network is randomly deployed, and the depth of the sensors can also be random. Therefore we need to run a few Bellhop simulations in order to get the average channel power gain as a function of distance for a source at constant depth and for several possible receiver depths. These simulations are carried out in the following way. First of all, to account for the uncertainty on the knowledge of the SSP, we generate several different random realizations of the SSP, starting from the one measured at the Trollet location in Figure 11. This SSP is sampled at several depths. To decrease the complexity of the computation, we first subsample it in such a way that its shape is preserved. The result of this subsampling can be seen in Fig. 88. Then we generate random SSP realizations from the subsampled SSP. Each realization is obtained by applying a different random displacement (drawn uniformly in the interval \([-4, +4]\) m/s) to each SSP sample. The displacement is sufficiently small to preserve the shape of the SSP. Then we proceed as follows:

1. We perform one Bellhop run for each randomized SSP in order to retrieve the channel power gain as a function of distance for every receiver depth; given that the SSP is slightly different, the realizations of the channel power gain will also be slightly different.
2. For each receiver depth:
   a. We collect the channel power gain as a function of distance from all Bellhop runs;
   b. We average over all Bellhop runs and thereby obtain an average channel power gain as a function of distance for every receiver depth;
Fig. 88: SSP at the Trollet location in the Trondheim fjord (solid black curve) and a sub-sampled version of the same SSP (dashed light grey curve) employed as a base for generating the random SSP realizations employed in Fig. 89.

Fig. 89: Channel power gain as a function of distance for various receiver depths in center of the CLAM test area in the Trondheim fjord. Each black-and-green curve corresponds to a different receiver depth. The solid light grey curve is point-wise average of all curves. The dashed grey curve is the channel power gain yielded by the Urick model in (9) for \( b = 1.865 \).

c. We draw the average computed at the previous step as a black-and-green curve in Fig. 89;

3. We compute the average of the channel power gain curves (across all depths) and plot it as a function of distance using a light grey solid curve in Fig. 89;

4. We compute the value of the path-loss exponent \( b \) in the Urick model (9), so that the curve of the channel power gain as a function of distance yielded by the Urick model best fits (in a least-squares sense) the average attenuation computed at the previous step. This value is found to be \( b = 1.865 \) for the scenario considered so far. The resulting Urick model curve is plotted as a grey dashed curve in Fig. 89.

The procedure outlined so far can be repeated for different sets of environmental parameters. For example, by employing the south-north profile in Fig. 14, it could be observed that the optimal value of the path-loss exponent is again \( b = 1.865 \). This allows us to run the simulations of UW-Polling using the empirical Urick model, with the value of the path-loss exponent above, and to conclude the simulations in a much lower time (and with equivalent results) than if we employed Bellhop.

The next subsection details and comments the simulation results.
7.9.3 Results

We recall that we employ scenarios 2 and 3 for the simulations of UW-Polling. In each case, a rectangular trajectory is set up for the AUV. The AUV follows the trajectory at a constant speed of 2.05 m/s, (4 knots), and keeps repeating the same route until the end of the simulation. For reference, the routes of the AUV in scenarios 2 and 3 are depicted in Figs. 90 and 91, respectively.

We start from Figs. 92(a) and (b), which show the throughput per node as a function of the packet generation rate \( \lambda \), for the three values of the data packet sizes. First of all we observe that the throughput per node in scenario 2 is equivalent to that of CSMA simulations in scenario 1 (Fig. 18) for all packet sizes. This is because in scenario 2 the area to be surveyed by the AUV and the number of nodes generating traffic are small enough to allow the protocol to reliably serve every node. This is confirmed by the PDR plot in Fig. 93(a), where the packet delivery ratio is on the order of 0.9 or higher for all packet sizes and for all values of \( \lambda \). Notably, the curves in Fig. 93(a) show generally better PDRs than the corresponding CSMA curves in Fig. 19. This is due to the poling mechanism, which avoids any multiple-access interference, so that the only sources of error are the noise, and the movement of
the AUV, which may have moved out of range when the last node in the polling list is about to transmit. The latter event is very unlikely, but it may take place in some cases.

The throughput of UW-Polling in scenario 3 (Fig. 18(b)) and the corresponding PDR in Fig. 93(b) show instead worse performance than CSMA's curves in Fig. 18. This is due to the larger number of nodes, and to the fact that the AUV takes more time to patrol the network and retrieve data from all nodes. This is especially true for 1000-byte packets, which take longer to transmit.

The delivery delay in Fig. 94 is instead much higher than the delay experienced with CSMA in scenario 1 (see Fig. 20). This is expected because the AUV moves along a predefined route, and any given node is only occasionally located within its communications range. The main component of the delay shown in Fig. 94 is in fact represented by the movement time of the AUV, and is expectedly larger in scenario 3 (Fig. 94(b)), where the trajectory of the AUV is almost twice as long, and the density of the network nodes per unit area is the same: in turn this means that there is a higher chance that a given node has to wait longer for the AUV to be available again.

The energy consumption for transmissions, receptions and idling is depicted in Figs. 95, 96 and 97, respectively. We notice that the amount of energy consumed in idling is about the same consumed by CSMA (see Fig. 23) in both scenario 2 and scenario 3. The transmit energy consumption is also very similar (compare Fig. 95(a), Fig. 95(b) and Fig. 21: it is slightly higher for UW-Polling because the sink takes active part in the administration of channel access by transmitting TRIGGER and POLL messages. Because channel access takes place in a controlled way, there is no substantial difference between the transmit energy consumption of UW-Polling in scenarios 2 and 3. The receive energy consumption is instead much higher in scenario 3, because the number of nodes in the network is double that of scenario 2 and almost all of them overhear the data transmissions meant for the AUV due to the low acquisition threshold set at the physical layer (see Section 4).

With the energy consumption values above, the worst-case node lifetime is about 22 days (assuming a worst-case energy consumption of 3 kJ for transmissions, 25 kJ for receptions and 5 kJ for idling). We note that this evaluation only applies to the static network nodes, as the main source of energy consumption for the AUV is likely going to be the propeller, not the communications subsystem.
Fig. 92: Throughput per node as a function of the packet generation rate per node for UW-Polling. (a) Scenario 2; (b) Scenario 3.

Fig. 93: Packet delivery ratio as a function of the packet generation rate per node for UW-Polling. (a) Scenario 2; (b) Scenario 3.

Fig. 94: End-to-end delivery delay as a function of the packet generation rate per node for UW-Polling. (a) Scenario 2; (b) Scenario 3.
Fig. 95: Energy spent for transmissions as a function of the packet generation rate per node for UW-Polling. (a) Scenario 2; (b) Scenario 3.

Fig. 96: Energy spent for receptions as a function of the packet generation rate per node for UW-Polling. (a) Scenario 2; (b) Scenario 3.

Fig. 97: Energy spent for idling as a function of the packet generation rate per node for UW-Polling. (a) Scenario 2; (b) Scenario 3.
7.9.4 Uw-Polling with AUVs moving at 8 knots

In this section we show the result of the simulation in the scenarios 2 and 3 (as in the previous simulations), but doubling the speed of the AUV. In this case, hence, the AUV patrols the network following the path depicted in Figs. 90 and 91 (for scenarios 2 and 3 respectively) at a speed of 4.10 m/s, (8 knots). As in the previous case, the AUV keeps following the same trajectory until the end of the simulation. The duration of the simulation is the same, hence the number of the laps done by the AUV is twice the number of laps considered in Section 7.9.3. The other simulations parameters (such as the carrier frequency, the payload size, etc.) are the same of the previous case. We would like to point out that in real world this speed would be unlikely for the AUVs. However, it may be interesting to see how UW-Polling reacts to the fact that the AUV doubles its speed and hence the nodes may be in coverage of the AUV for a lower amount of time.

In the following pictures, the simulations results with the AUV travelling at 4 knots are depicted with a black line, while the new results (with AUV travelling at 8 knots) are depicted using a red line.

Looking at the various metrics plotted in the scenario 2, in the left column of Figs. 98, 99, 100, 101, 102 and 103, we can say that change the speed of the AUV has no effect on the performance of the protocol. This is because this particular topology in scenario 2 is such that it takes several minutes for a node to exit the AUV’s coverage range; in fact, the sensors topology in scenario 2 is not so wide. A sensor, for these reasons, is in the AUV’s coverage range for an amount of time that is enough for transmitting its own packets, even if the AUV moves at 8 knots. In many cases (i.e., at the throughput in the case of packets dimension of 100 byte in Fig. 98a) the two curves are perfectly overlapping. Moreover, there are very small differences in terms of throughput, packet delivery ratio and delay.

Taking a look to the results of the simulation on scenario 3, instead, there are bigger differences, especially in terms of Packet Delivery Ratio (Fig. 99b); in the scenario 3 the sensors are deployed in an area wider than the one in scenario 2. Hence, in this case it is more likely that a node exits the AUV’s coverage range during the transmission of its packets. This fact reflects on a lower throughput and a higher Packet Delivery Ratio, especially when the traffic increases, as we observed in the previous section (Figs. 92b and 93b). Doubling the speed of the AUV, this effect is amplified. In fact, with the AUV moving twice as fast, a node has less time for transmits its packets (roughly we can say that it has half the time for transmit its packet that it has in the case of the AUV moving at 4 knots). This reflects on a lower packet delivery ratio. In particular, we can observe how the difference between the two cases increases as the traffic increases: a node generates a packet more often, so the probability that a packet is dropped (either because the node queue is full or because a packet is transmitted when the node is about to exit the coverage range of the AUV) increases. The same considerations can be drawn for the throughput (Fig. 98b), even if the differences are not as significant as in the case of the Packet Delivery Ratio. As a straightforward consequence, also the delivery delay increases.
(Fig. 100b): a node has less time to deliver a packet, so it may happen that a node stores its packets in the queue for more time.
Fig. 98: Throughput per node as a function of the packet generation rate per node for UW-Polling. The red line represent the case with the AUV travelling at 8 knots, while the black line represent the AUV travelling at 4 knots. (a) Scenario 2; (b) Scenario 3.

Fig. 99: Packet delivery ratio as a function of the packet generation rate per node for UW-Polling. The red line represent the case with the AUV travelling at 8 knots, while the black line represent the AUV travelling at 4 knots. (a) Scenario 2; (b) Scenario 3.
Fig. 100: End-to-end delivery delay as a function of the packet generation rate per node for UW-Polling. The red line represents the case with the AUV travelling at 8 knots, while the black line represents the AUV travelling at 4 knots. (a) Scenario 2; (b) Scenario 3.
Fig. 101: Energy spent for transmissions as a function of the packet generation rate per node for UW-Polling. The red line represent the case with the AUV travelling at 8 knots, while the black line represent the AUV travelling at 4 knots. (a) Scenario 2; (b) Scenario 3.

Fig. 102: Energy spent for receptions as a function of the packet generation rate per node for UW-Polling. The red line represent the case with the AUV travelling at 8 knots, while the black line represent the AUV travelling at 4 knots. (a) Scenario 2; (b) Scenario 3.
Fig. 103: Energy spent for idling as a function of the packet generation rate per node for UW-Polling. The red line represent the case with the AUV travelling at 8 knots, while the black line represent the AUV travelling at 4 knots. (a) Scenario 2; (b) Scenario 3.
7.9.5 Concluding remarks on UW-Polling simulations

The results in this section suggest that UW-Polling is a valid alternative for data retrieval in underwater networks, although more expensive from an architectural point of view, as it involves a mobile sink (i.e., the AUV) instead of a static sink. Nevertheless, UW-Polling can effectively administer the data retrieval process, and has been designed to natively give priority to data of interest. In order to be uniform with the simulations of the other protocols in this deliverable we have not tested UW-Polling with different types of traffic (e.g., alarm traffic vs. background monitoring traffic), hence the “high priority” data in our simulations is represented by the most recent sensor readings generated by a node. The simulation of UW-Polling with different definitions of priority is left for possible inclusion in D4.3.

We remark that the energy consumption induced by UW-Polling is dominated by the reception component. Hence, we infer that it would be very beneficial for a node to be able to early terminate receptions of data packets not meant for itself, before actually having to conclude the packet detection process. This conclusion is in line with the remarks in Section 7.8.

As a final note, we observe that doubling the speed of the AUV from 4 to 8 knots does not lead to significant differences in the performance of UW-Polling, with the exception of a slightly lower PDR in the 8 knots case.

8 Routing protocol simulations

8.1 SUN

We start by discussing the results obtained from the simulation of the SUN protocol described in Section 3.1. The results below are provided in pairs: in each pair of figures, the one on the left refers to scenario 2 (19 nodes plus the sink), whereas the one on the right refers to scenario 3 (39 nodes plus the sink). We recall that SUN will be run using the hop count metric, and that the routes are configured not to expire for the whole duration of the simulation since the network and the environment are both static. In any event, the routes can still be declared invalid due to the loss of packets on one of their links.

Fig. 104 depicts the throughput per node as a function of the traffic generation rate \( \lambda \). We observe that the throughput increases almost linearly with \( \lambda \) for \( \lambda \leq 0.05 \) packets per minute in scenario 2, and for \( \lambda \leq 0.025 \) packets per minute in scenario 3. The main reason of this difference is the heavier interference suffered by the nodes in scenario 3, as the amount of traffic to be delivered to the sink is twice the traffic in scenario 2. However, it should be noted that the throughput in Fig. 104(a) remains stable for increasing \( \lambda \), whereas the one in Fig. 104(b) not only reaches a lower maximum value than in Fig. 104(a), it also decreases steadily at a rate that is higher for larger packet sizes. The reason is to be looked for in the path search...
procedures of SUN. In particular, for 20 nodes, the amount of traffic is still sufficiently limited to allow route search procedures to be completed correctly. On the contrary, in scenario 3 the traffic is very intense, and causes sufficient interference to disrupt route search procedures. Hence, when a route is declared invalid (e.g., due to an excessive number of transmission errors over a given link), and a new route is sought, the signaling packets transmitted for this purpose may be lost, and the route never re-created. Also, a node with an invalid route will notify the nodes upstream, thereby injecting even more signaling traffic in the network. As a consequence of all facts above, the communications will be impaired, and only the nodes closest to the sink will be able to actually deliver some data packets.

This fact is better seen from Fig. 105, which depicts the throughput per node as a 3d histogram, where each bar is placed at the center of one of the grid elements in Fig. 9 and represents the throughput of the node placed in that grid element. The graph is depicted for $\lambda = 0.06$ and for a packet size of 500 bytes (these values allow us to investigate the behavior of the network under very high load, as seen from the throughput curves in Fig. 104(b)). We observe that the highest throughput is reached near the sink (upper left corner of the picture); on the contrary, the nodes farthest from the sink (lower-right corner) experience bad interference conditions, and can only rarely find a good route to transmit their own data; hence, they experience lower throughput. It should also be noted that the nodes located in the central grid line (light grey bars) are interfered by the nodes to the sides of the network (dark grey and white bars), which in turn experience reduced interference from each other due to their larger mutual distance. The net effect is the throughput experienced by the nodes in the central line is lower than that of the external lines.

The discussion above is further supported by Fig. 106, where we show that the PDR decreases for increasing $\lambda$. This is expected from the throughput curves above. As expected, smaller packet sizes lead to higher packet delivery ratios. However, the throughput is correspondingly lower, as depicted in Fig. 104. We note that the packet losses due to failed retransmissions are not the only reason why the PDR is less than 1. Packets may also be discarded because the routes are declared invalid (e.g., due to previous problems in the forwarding process), and the nodes have troubles finding a new valid route: in this case, newly generated packets would be put in the buffer, or dropped if the buffer is found already full. This event takes place quite often, and is the major source of packet losses. This is confirmed in Fig. 107, where the number of link-level retransmissions per packet is always very low. Notably, this already low number decreases with increasing $\lambda$, in the most critical situation (packets of 1000 bytes in Fig. 107(b)), meaning that the route discovery procedure is rarely completed correctly because of heavy interference and contention.

The observation above is also in line with Fig. 108, where the delay per km is expected to be very high both in scenario 2 and in scenario 3, with acceptable values found only for low values of $\lambda$. The decreasing behavior of the delay curves for high values of $\lambda$ in Fig. 108(b) is due to the very low PDR. In fact, when most nodes in the network do not have a valid route, the nodes closer to the sink tend to seize the channel. As these nodes do not receive interference from farther nodes any more, their transmissions are likely to be successful and to take less time, hence
the lower average end-to-end delay per unit length. The distribution of the delay in scenario 3 as a function of the position of the node in the grid (see Fig. 9 for reference) is shown in Fig. 109, for $\lambda = 0.06$ packets per minute per node and 500-byte packets. We recall that this is a saturation condition for SUN (see Fig. 104(b)). We observe that only the nodes closest to the sink (upper-left corner of the figure) experience acceptable delays. The other nodes can still be served by the network, but with much longer delays, that increase for increasing distance from the sink.\footnote{The performance of SUN in networks subject to lower traffic demands (e.g., due to a lower local density of nodes) is much better. Such performance will be seen, e.g., in D5.2, where a likely deployment for the final CLAM demonstration will be simulated. In the present deliverable, we aim instead at measuring the performance of SUN both under low traffic requirements and under stress conditions, with slightly more emphasis to the latter, as the former will be more widely described in the contents of D5.2.}

Along the same line, Fig. 110 shows that the overhead of the protocol increases for increasing $\lambda$. This is a further proof that, at high traffic generation rates, the protocol tries to find routes more often. In turn, this depends on the fact that the routes become invalid due to repeated errors. From Fig. 110 we can also verify the expected result that the protocol overhead ratio is larger for smaller data packets, as long as the network is not congested. Since the level of interference increases and the nodes fail to find valid routes (as described above), the consequent decrease of data packet transmissions makes the overhead for 1000-byte packets equivalent to that of 100-byte packets. This can be seen in Fig. 110(b).

To conclude the evaluation of SUN, we show in Figs. 111 to 113 the energy consumption per node, by separating the contributions due to transmissions, receptions and idling, as was done for the analysis of MAC protocols. The results confirm the analysis above. In particular, the fact that the number of transmissions decreases due to the lack of valid routes is reflected in both the transmit and the receive energy consumption (Figs. 111 and 112, respectively), which decrease for increasing $\lambda$, for all packet sizes. The energy spent for idling, instead, expectedly decreases in a manner that is inversely proportional to $\lambda$. (Fig. 113) and is an important contribution to energy consumption. To better highlight the balance between the energy consumption for transmissions, receptions and idling, we show in Fig. 114 the normalized energy consumption per node as a function of the node position in the grid depicted in Fig. 9. We recall that the normalized energy consumption per node is defined as the ratio of the total energy spent by a node during a simulation, divided by the energy the node would spend if it stays idle for the whole simulation time. When this ratio is high, idling is a minor reason of energy consumption. Once again, we set $\lambda = 0.06$ and we consider 500-byte packets. From Fig. 114, we observe that the normalized energy consumption is steadily around 7 for all nodes (or, in other words, idling counts for 1/7th of the node energy consumption, on average). Hence, there would be little point in endowing the nodes with duty cycling procedures, as this would allow to reduce only the idling energy component. Moreover, duty cycling would require to set up signaling procedures to coordinate the nodes, making it more unclear what the real energy gain would be. This is in line with the observations in [4]. Instead, it would be meaningful to find ways to reduce the energy
consumption during reception, while still keeping the nodes awake to be aware of the status of the network. For example, this may be achieved by making modems aware of the header of a packet, and quit reception as soon as the packet is declared “uninteresting” from the upper layers. In turn, this would require real-time interactions among the physical layer and the networking protocols, and is thus left as a future research direction.

From the energy data above, the expected worst-case node lifetime with SUN is approximately 18 days in scenario 2 (assuming a worst-case energy consumption of about 8 kJ for transmissions, 30 kJ for receptions and 2 kJ for idling); this lifetime is reduced to approximately 6 days in scenario 3 (assuming a worst-case energy consumption of about 14 kJ for transmissions, 110 kJ for receptions and 2 kJ for idling).

Concluding remarks — In this section, the SUN protocol for underwater networks was evaluated in two scenarios relevant for CLAM. These scenarios have been considered, on one hand, to adhere to prospective network deployments in the proximity of oil wells, and on the other hand to originate sufficient traffic and interference to put the network under stress, and thereby measure when the network starts giving undesirable performance. We observed that SUN behaves well until the traffic generation rate is too high for the packets to be delivered reliably, and quantified the maximum rate by means of the throughput plots in Figs. 104(a) and 104(b), which refer to a network of 20 nodes and a network of 40 nodes, respectively. We also observed that the energy consumption is dominated by the reception component, while idling counts for about 1/7 of the overall energy consumption per node. For this reason we concluded that reducing the receive energy consumption (e.g., by early termination of receptions upon reading of the header contents) would be more meaningful than setting up some form of duty cycling.

The simulation of the protocol in a deployment that will be likely employed in the final CLAM demonstration is contained in D5.2, and shows that the protocol offers indeed very good performance, due to the lower network density, which translates into lower interference by neighboring nodes, and hence fewer route error messages. As a final remark, we note that SUN would perform much better if the parameters of the protocol are adapted to the scenario. This will be shown in D5.2, where changing the timings of the agent controlling the buffer of a node has a significantly beneficial impact on the throughput and delay of the network.
Fig. 104: Throughput per node as a function of the packet generation rate per node for SUN. (a) Scenario 2; (b) Scenario 3.

Fig. 105: Throughput as a function of the position of the nodes in the grid shown in Figure 9, for $\lambda = 0.06$ packets per minute per node, and a packet size of 500 bytes. The sink is placed at the position (0.5 km, 0.5 km), 10 m below the surface.
Fig. 106: Packet delivery ratio as a function of the packet generation rate per node for SUN. (a) Scenario 2; (b) Scenario 3.

Fig. 107: Number of link-level retransmissions per packet as a function of the packet generation rate per node for SUN. (a) Scenario 2; (b) Scenario 3.

Fig. 108: End-to-end delivery delay as a function of the packet generation rate per node for SUN. (a) Scenario 2; (b) Scenario 3.
Fig. 109: End-to-end delivery delay per node as a function of the position of the nodes in the grid shown in Figure 9, for $\lambda = 0.06$ packets per minute per node, and a packet size of 500 bytes. The sink is placed at the position (0.5 km, 0.5 km), 10 m below the surface.

Fig. 110: Protocol overhead ratio as a function of the packet generation rate per node for SUN. (a) Scenario 2; (b) Scenario 3.
Fig. 111: Energy spent for transmissions as a function of the packet generation rate per node for SUN. (a) Scenario 2; (b) Scenario 3.

Fig. 112: Energy spent for receptions as a function of the packet generation rate per node for SUN. (a) Scenario 2; (b) Scenario 3.
Fig. 113: Energy spent for idling as a function of the packet generation rate per node for SUN. (a) Scenario 2; (b) Scenario 3.

Fig. 114: Normalized energy consumption per node (defined as the total consumed energy per node divided by the energy a node would spend if it remained idle for the whole simulation time) as a function of the position of the nodes in the grid shown in Figure 9, for $\lambda = 0.06$ packets per minute per node, and a packet size of 500 bytes. The sink is placed at the position (0.5 km, 0.5 km), 10 m below the surface.
We now present the results of the CARP protocol simulations in scenarios 2 and 3.

8.2.1 Protocol parameters setting

Considering the acoustic modem parameters setting described in Section 4:

- Carrier frequency: 25.6kHz
- Bandwidth: 4kHz
- Source Power Level (SPL) at 1 m: 178dB re µPa
- Modulation: BPSK

the source power level used for PING and PONG packets has been set to 175dB re µPa (6W). This value allows CARP to select node relays with robust links for both control and data packets assuming a maximum payload of 1000 bytes (the one considered for this performance evaluation).

The sizes of PING and PONG packets are set to 5B, the ACK packets are set to 4B, while the HELLO packets are 3B long. The CARP MAC header is 4B long.

8.2.2 Results

Figure 115 shows the packet delivery ratio of CARP for the three considered payload sizes. In all cases, as the traffic load increases the packet delivery ratio decreases. This effect is due to multiple reasons. When the number of packets is higher, the number of times the nodes find the channel busy increases. Moreover, and more significantly, with more packets the chances of collisions are higher, and the corresponding re-transmissions degrade the packet delivery performance. This is especially true in a multi-hop scenario since each hop generates extra data packets, new overhead (control packets) and collisions also happen because of interference generated by transmitting nodes multiple hops away.

This behavior is more evident comparing the results for scenario 2 (Figure 115a) with that of scenario 3 (Figure 115b). In scenario 3 we have twice the number of nodes than in scenario 2, deployed in an area which is twice larger with respect to scenario 2. For the same offered load values, twice more packets are generated in the network which can traverse longer routes towards the sink, thus resulting in a higher number of nodes competing to reserve the channel at the same time.

As expected the increased number of retransmissions (Figure 116) and of unsuccessful channel access translate into higher latency at high load. This is clearly shown in Figure 117.

It is interesting to evaluate the impact of different payload sizes on CARP performance. Results show that longer payloads result into lower packet delivery ratio and higher normalized latency, as it translates into a higher traffic load injected in the network.
On the other hand, using longer payloads reduces the overhead the protocol pays in terms of delay and control information needed to transmit the same amount of data achieving a higher throughput (Figure 115) and a better channel utilization. Figure 119 presents CARP overhead ratio. As expected a higher overhead is paid when shorter data payloads are used for the same reasons discussed above. However, it is interesting to notice that such ratio is really stable when $\lambda$ increases. This means that the number of control packets transmitted to reserve the channel is always proportional to the number of transmitted data packets. When the offered load increases and a lower number of packets is correctly delivered to the sink (Figure 115), CARP does not start pushing more control packets into the channel.

Node energy consumption is presented in Figure 120. The time each node takes to transmit on average 100 packets is inversely proportional to the network offered load. Lower packet generation rates result in longer amount of times node spend transmitting packets. Instead, when the offered load is low, the time each node spends in idle listening is much longer than the time it actually spends transmitting and receiving packets. In this case the idle time strongly affects the node en-
ergy consumption performance. When the offered load increases, similar values are experienced for the node energy consumption related to packet transmission and reception with a great reduction in the energy spent in idle. This clearly show how the node idle time cannot be ignored when planning a sea trial and has to be carefully taken into account. In scenarios with longer data payloads, higher energy consumption values are obtained since a higher energy is needed for each data transmission. Using a lower transmission power when transmitting PING and PONG packets is not only an effective solution to select relays which are connected to the source by robust links but is also able to significantly reduce (40%) the energy consumption associated to control packets transmissions.

We now analyze how the performance discussed above are affected by the source and relay nodes. In each considered scenario, the network area is subdivided into elements of roughly equal size (3 × 6 elements in scenario 2 and 3 × 12 elements in scenario 3), and nodes are randomly placed within one of these elements. For our performance evaluation different realizations of these random topologies have been considered but a node with the same ID is always placed within the same grid.
element. This allow us to present a three-dimensional graph showing the spatial distribution of a selection of the considered metrics (Figures 121 to 122). We consider scenario 3, focusing on nodes with ID ranging from 1 to 36, with $\lambda = 0.06$ packets per minute per node and the data packet length set to 500 bytes. The sink has node ID 0, and is located 0.5km towards North and 0.5km towards East.

Figure 121 shows a fair distribution of the throughput performance of nodes located in different grid elements. Nodes closer to the sink need a lower number of hops to correctly deliver their data to the it but at the same time have a higher probability to be selected as relays by other nodes, resulting in a higher local traffic load. The same happens for nodes in the middle of the target area (2km towards North). Having nodes far away from the sink reaching a throughput similar to the ones in the sink area shows how good is CARP in delivering data in a multi-hop network.

A similar fair distribution of the normalized energy consumption is presented in Figure 122. Again nodes in the sink area and in the middle of the target area incur a higher number of transmission and reception (as they overhear a higher number of packets due to their location) thus resulting in a higher energy consumption.
However, as for the throughput results, the overall energy consumption is really similar for nodes in different grid elements, showing the efficiency of CARP in distributing the protocol load and energy consumption among different nodes. The residual energy of a node is one of the key metric which is used by CARP to select the best relay node. When a node starts reducing its energy level other nodes are preferred to act as relays.

End-to-end delivery delay is presented in Figure 123. This time, nodes close to the sink area have to traverse a lower number of hops to reach the sink resulting in shorter delay. The data packet latency increases when moving far away from the sink.

9 Optimal Periodic Joint Routing and Scheduling

In this section we study the joint optimal routing/scheduling problem for underwater sensor networks with periodic traffic. Joint routing/scheduling solutions for wireless networks have been widely investigated in the literature. In general, determining a feasible routing scheme together with an overall transmission schedule, is an interesting and challenging problem [42, 43]. In underwater networks the problem is
exacerbated because of the more stringent delay and interference constraints and new solutions must be devised possibly using different approaches. In fact, when studying terrestrial radio network the link propagation delay is usually assumed to be zero, but this assumption becomes inadequate for underwater acoustic network where the propagation speed of acoustic signals (around 1500 m/s) is roughly five orders of magnitude slower than that of RF signals. This is usually called “Spatial-Temporal Uncertainty” [30]. Moreover, considering all the differences between RF communication in air and acoustic transmission underwater, solutions for terrestrial wireless (sensor) networks cannot be directly applied to Underwater Wireless Sensor Networks (UWSNs) for underwater acoustic communications the signal power attenuation is lower than the one terrestrial radio networks, which, combined with the long propagation delays, produces an interference range potentially much longer than the transmission one [27, 44].

We propose a new optimization framework for underwater sensor networks. We formulate the routing/scheduling problem into an Integer Linear Programming (ILP) model which yields the joint optimal routing and scheduling which maximizes the network throughput and minimizes the energy consumption. Differently from most of prior research in the area, we consider an accurate interference model where we account not only the presence of other single transmitters but also the combination
of multiple transmissions that can overlap during the reception at the packet destination.

Unfortunately, the resulting solution is NP-complete, and thus it is not practical for large topologies. To overcome this limitation, we also propose a heuristic solution which scales well with the problem size. In the heuristic, since the complexity of the optimal strategy stems from the need to jointly optimize routing and scheduling, we use the “divide et impera” principle: first we compute the routing; then, with the routing fixed, we schedule packets transmissions. We determine the routing by solving a suitable and efficient optimization problem. More precisely, we consider a simple fluid flow model approximation of the network traffic and determine the routing which minimizes our joint routing-scheduling objective function. Given the routing, then we compute the actual data packet schedule. Our scheduling heuristic is motivated from the observation that the scheduling problem can be regarded as a specialized vertex-coloring problem of a labeled network conflict graph where labels represent the relative delay which cause a conflict between links transmissions. The solution is thus inspired by simple greedy vertex-coloring heuristics which have been adapted to account for the conflicts delay expressed via the edge labels. The proposed heuristic is able to scale to larger network producing results really close to the optimal ones.
The solution produced by the ILP model and the heuristic can be used as guideline and benchmark for the design and evaluation of distributed protocol stack.

The remainder of the section is organized as follows. In Section 9.1 we present related results in the literature. In Section 9.2 we present the system model. In Section 9.3 we formulate the ILP model for joint routing-scheduling optimization.

9.1 Related work

In literature many studies have addressed optimal solutions for UWSNs. Several mathematical models have been presented for optimal node placement [29, 45–47] also considering the possibility of varying/adjusting the sensor depth to dynamically adapt to underwater dynamics [48, 49]. Evolutionary algorithms have also been proposed [50]. In [51] an optimal MAC, named ST-MAC, scheduling problem is formulated using a weighted, directed conflict graph with the aim of minimizing the frame size while archiving the maximal throughput for one-hop acoustic network. ST-MAC schedules transmissions by assigning a color (an integer) to each edge in the conflict graph. Unlike the traditional vertex-coloring problem, the difference of colors between adjacent vertices must be larger than the weight of edge between the corresponding vertices in the conflict graph. Therefore, existing heuristics cannot be easily applied to this formulation. Moreover, in order to make vertex-coloring algorithms work, they have to divide transmissions into multiple unit size slots, and force the weight of edges of a conflict graph to be integer. In [52] another MAC scheduling in one-hop acoustic networks is proposed. The scheduling problem is reduced to the standard TSP so that existing heuristics can be directly applied. In this case, the frame is not divided into multiple equal-size time slots so that scheduled transmissions can start at any time (instead of only on the boundary of slots), which improves channel utilization as well as network throughput. Both these solutions address only one-hop acoustic networks, assuming that every time two transmissions overlap at the receiver a collision occur. There is no investigation about the use of an underwater acoustic channel attenuation model and about the presence of multiple interfering nodes. Necessary and sufficient condition for the correct delivery of a data packet is that this is the only transmission which takes place in the transmitters and receivers radio ranges. In UWSNs a more accurate interference model should be assumed, considering that multiple transmissions, occurring also in different times, can overlap during the reception at the packet destination producing a collision. A jointly optimization of placement, scheduling and routing is presented in [53] with the goal to minimize energy consumption, modeling the behavior of the network during a large single frame. The paper investigate small-scale multi-hop networks considering the presence of multiple interfering nodes and the use of an underwater acoustic channel attenuation model, which is based on empirical formulas derived from [22]. Although this is work is more accurate and complete than the previous ones, the model formulation is quite complicated and can be applied
only to small-scale networks. Moreover, a more realistic and accurate underwater acoustic attenuation model should be used with respect to empirical formulas.

In our work, similarly to [55], we jointly address the routing/scheduling problem. We focus on multi-hop networks, considering an accurate interference model and assuming as channel attenuation model the Bellhop ray tracer [17]. Bellhop is used with historical environmental data and provides us with an accurate description of the underwater acoustic channel behavior. We have also investigated a heuristic solution to scale to larger networks, using the “divide et impera” approach: we first compute the routing using a fluid flow model; then, with the routing fixed, we schedule data packets transmissions. The proposed heuristic solution is able to scale to larger network producing results really close to the optimal ones.

9.2 System Model

9.2.1 Network Model

We model the network with a directed graph $G(V,E)$, where $V$ is the set of sensor nodes and $E$ is the set of links available in the network. Each node is located in a three-dimensional space modeling the underwater deployment area and transmits at a fixed power level $P$. There is a link from node $u$ to $v$ if a reliable transmission can occur between the two nodes, i.e. if the Signal-to-Noise Ratio (SNR) of $u$’s transmission at $v$ is higher than a given threshold $SNR_{th}$. Given the carrier frequency $f$, for each pair of nodes $(u,v)$ we compute the transmission gain $Gain(u,v)$. Then $(u,v) \in E$ if and only if $\frac{P \cdot Gain(u,v)}{N} \geq SNR_{th}$, where $N = N(f) \cdot \Delta f$ is the ambient noise power. $N(f)$ is the noise power spectral density as described in [22]; $\Delta f$ is the receiver noise bandwidth (a narrow band around the frequency $f$). The propagation delay $d(u,v)$ between $u$ and $v$ is expressed in number of slots and it is obtained by the link length divided by the product of sound speed (we assume a sound speed in water of 1500m/s) and the duration of the nominal time slot in seconds (slot duration). For each node $v$, we also denote with $E^{out}(v)$ the set of outgoing links from $v$, and with $E^{in}(v)$ the set of links entering $v$.

9.2.2 Traffic Model

We consider a scenario where a sink node is in charge of collecting all the information generated from the sensors. We will denote by $g(v,t)$ the number of packets generated by node $v$ at time $t$. Packets which cannot be transmitted directly to the sink are relayed through intermediate nodes. Based on the fact that sensor traffic for underwater monitoring is expected to be highly periodic, we consider a periodic scheduling of transmissions from the nodes. The fundamental period, called frame, is divided into multiple slots each of a fixed length, representing the time needed to transmit a data packet which we assume equal for all nodes. While a data packet
has to be transmitted at the beginning of a slot, receptions can happen at any time according to transmitter/receiver propagation delay.

9.2.3 Conflict and Interference Model

Link conflicts/interference reduce the number of transmission opportunities nodes have to transmit packets. Conflicts occur simply because a node cannot receive and transmit at the same time or because a node cannot receive from two distinct transmitters at the same time. Interference conflicts occur when the transmission over a link prevents correct reception over another link. In the literature, most of the optimality frameworks for scheduling/routing are based on simplified interference assumptions, where necessary and sufficient conditions for the correct delivery of a packet are that this is the only transmission which takes place in the transmitters and receivers radio ranges. Basically, this means that both hidden and exposed terminal problems must be avoided. This leads to developing nice and clean mathematical constraints, which are however not suitable for the underwater scenario, given that destructive interference is typically due to multiple far away transmissions [44].

Differently from these approaches, we consider a complete interference model which takes into account not only the presence of other single transmitters but also the combination of multiple transmissions that can overlap during the reception at the packet destination. We want to avoid that the received Signal-to-Interference Ratio (SIR) is too low for some received packets, e.g., since many far-away nodes are all transmitting (simultaneously or at different times) with the result that the overlapping receptions at some receiver impair correct reception.

We classify conflicts in two general categories:

a) duplex conflict: Since we assumed half-duplex communication, a node cannot receive packets from more than one link at time (Figure 124c), and it cannot transmit simultaneously on more than one link (Figure 124a). Moreover, it cannot transmit while it has not completely received a packet (Figure 124b).

b) interference conflict: This conflict occurs when a concurrent transmission on link \( f \) may disturb the reception on link \( e \) (Figure 124d). If the interference generated by transmission on \( f \) is too powerful, the packet transmitted on link \( e \) cannot be correctly received.

Let us denote with \( I(e) \) the set of possible interfering links with link \( e \).

\[
\forall f \in E, f \in I(e) \iff \frac{P \cdot \text{Gain}(e, \text{src}, \text{dst})}{P \cdot \text{Gain}(f, \text{src}, \text{dst}) + N} < \text{SIR}_{th}
\]

where \( \text{Gain}(e, \text{src}, \text{dst}) \) is the gain of signal transmitted on link \( e \), \( \text{Gain}(f, \text{src}, \text{dst}) \) is the gain of signal transmitted by the source of link \( f \) and heard by destination of link \( e \), and \( P \) is the transmission power that we assume equal for all nodes. The expression can be easily generalized to the case of multiple interferer links (Equation 18).
9.3 Optimization Model

Given the network model $G(V,E)$, we consider periodic sensors traffic generation. Our goal is to determine the minimum frame length $T$ (thus maximizing network throughput) and the associated transmission schedule - which avoids conflicts among link transmissions - while minimizing the overall energy consumption. A schedule can be specified as a sequence $S_t$, $t = 0, 1, \ldots, T - 1$, of sets of links, that may transmit simultaneously without conflicts. In solving the problem we exploit the periodic network behavior, and study a solution where the packet generation scheme, routing and scheduling are repeated through frames. Under these assumptions an optimal solution for a single frame, necessarily results in an optimal solution over the lifetime of the network.

We found that the joint determination of the frame length $T$ and the associated periodic schedule is an extremely difficult problem, complicated by the fact that $T$ should be both a variable and a parameter to be accounted for when writing the conflicts/interference constraints (see below). To overcome this difficulty we tackle the computation of the frame length $T$ and the associated periodic transmission schedule iteratively. The idea is simply to solve the joint periodic routing schedule problem for a fixed frame length $T$ and then to carry out a binary search to determine the optimal schedule length $T^*$. 

---

**Fig. 124: Conflicts**

(a) transmission-transmission conflict

(b) transmission-reception conflict

(c) reception-reception conflict

(d) interference conflict
Let $\text{OPT}(T)$ denote the optimal joint periodic routing schedule problem for a fixed frame length $T$ and let $S = S(T)$ denote the associated optimal schedule (under the assumption that $\text{OPT}(T)$ has a feasible solution). Then the overall optimal schedule $S^*$ is $S^* = S(T^*)$ where $T^* = \arg\min_T S(T)$ is the minimum frame length for which a feasible solution exists. $\text{OPT}(T)$ can be formulated as a centralized ILP problem as follows.

### Table 2: Notation Summary

| $v, u$ | nodes of network graph |
| $e, f$ | links of network graph |
| $e_{\text{src}}, e_{\text{dst}}$ | source node, and destination node of link $e$ |
| $E^\text{out}(v), E^\text{in}(v)$ | sets of outgoing/incoming links from node $v$ |
| $I(e)$ | set of possible interfering links with link $e$. |
| $d(u, v)$ | propagation delay expressed in number of slots between node $u$ and $v$ |
| $d_e$ | propagation delay expressed in number of slots on link $e$. $d_e = d(e_{\text{src}}, e_{\text{dst}})$ |
| $d_{f,e}$ | propagation delay expressed in number of slots from the source of link $f$ to the destination of link $e$. $d_{f,e} = d(f_{\text{src}}, e_{\text{dst}})$ |
| $T$ | frame size |
| $t_e, t_f$ | transmission time on link $e$, and link $f$ |
| $L$ | number of frames to look for possible transmissions conflicts |
| $P$ | transmission power |
| $N$ | ambient noise power |
| $\text{Gain}(u, v)$ | gain of signal transmitted from node $u$ to node $v$ |
| $\text{SNR}_{th}$ | Signal-to-Noise Ratio (SNR) threshold |
| $\text{SIR}_{th}$ | Signal-to-Interference Ratio (SIR) threshold |
| $g(v, t)$ | packets generated by node $v$ at time $t$ |
| $X(e, t)$ | binary variable equal to 1 if link $e$ is active during time $t$ |
| $B\text{In}(v, t)$ | integer variable that counts the number of packets added to node $v$ buffer at time $t$ |
| $B\text{Out}(v, t)$ | integer variable that counts the number of packets going out from node $v$ at time $t$ |
| $B\text{Size}(v, t)$ | size of buffer of node $v$ at time $t$ |

**Variables**

- $X(e, t)$: binary variable equal to 1 if link $e$ is active during the time slot $t$;
- $B\text{In}(v, t)$: integer variable that counts the number of packets added to node $v$ buffer at the beginning of time slot $t$;
Objective Function

Our goal is to minimize the overall energy consumption. The objective function is:

$$\min \sum_{e \in E} \sum_{t=0}^{T-1} X(e,t) \cdot E_{tx}$$  (11)

where $E_{tx} = P \cdot \text{slot duration}$ is the energy consumption for each transmission. Remember that we iteratively search for the minimum frame length $T$, which results in a feasible solution for the ILP problem, in order to maximize the network throughput.

Constraints

Sink constraints ((24)-(25))

As the sink node is only a data collector, it cannot generate traffic or transmit data packets. The following constraints are used to inhibit generation and transmission of data packets from the sink node.

Inhibition sink generation, and transmission:

$$\sum_{t=0}^{T-1} g(\text{sink}, t) = 0, \sum_{e \in E^{out}(\text{sink})} \sum_{t=0}^{T-1} X(e,t) = 0$$  (12)

Conflict Constraints ((26)-(29))

Conflicts among link transmissions reduce the transmission opportunities. This is captured by the following sets of constraints

• concurrent transmission conflict (26): This constraint avoids concurrent transmission over two links going out from node $v$ (Fig. 124a). As we assume that each transmission start at the beginning of a slot, we ensure that at most one link is active on $E^{out}(v)$ for all possible slot $t \in [0, T - 1]$, where $E^{out}(v)$ is the set of links coming out from node $v$. Hence:
\[ \sum_{e \in E^{in}(v)} X(e,t) \leq 1 \forall v \in V, \forall t \in [0,T-1] \quad (13) \]

- **concurrent reception conflict** (27): These constraints prevent that a node \( v \) can receive over two different incoming links \( e, f \in E^{in}(v) \) at the same time (Figure 124c). Assume a transmission over link \( e \) starts at time \( t_e \) and a transmission over link \( f \) starts at time \( t_f \). A conflict occurs whenever the reception of the two packets overlaps in time. Let \( d_e = d(e, \text{src}, e, \text{dst}) \) and \( d_f = d(f, \text{src}, f, \text{dst}) \) denote the propagation delay (always measured in number of slots) along links \( e \) and \( f \). Then a conflict occurs whenever the two packet reception time intervals \([t_e + d_e, t_e + d_e + 1]\) and \([t_f + d_f, t_f + d_f + 1]\) are not disjoint (Figure 125a, 125b). It is easy to verify that this corresponds to the constraint

\[ \forall e, f \in E^{in}(v), \forall t_e, t_f \in [0,T-1] \]

\[ X(e,t_e) + X(f,t_f) \leq 1 \iff t_e + d_e - 1 < t_f + d_f < t_e + d_e + 1. \quad (14) \]

In addition, we need to consider the case where conflicts occur among transmissions originated in different frames (Figure 125c). Indeed, it is possible that \( t_f + d_f > T \), since we impose no constraint on the ending of the reception of a transmitted signal within the end of the frame in which transmission started. Hence conflicts may be caused by transmission occurred in previous frames. To capture this, we move an imaginary window of length \( T \) through \( L \) consecutive frames, where \( L \) is the maximum number of frames a packet will take to reach the farthest distance within the network, i.e. it is the minimum number of frames greater or equal to the maximum propagation delay and generalize the constraint (14) as follows:

\[ \forall v \in V, \forall e, f \in E^{in}(v), \forall t_e, t_f \in [0,T-1], \forall k \in [0,L] \]

\[ X(e,t_e) + X(f,t_f) \leq 1 \iff t_e + d_e - 1 < t_f + d_f - kT < t_e + d_e + 1 \quad (15) \]

where the term \( kT, k \in [0,L] \), allows us to account for conflicts between transmissions occurring in different frames.

- **transmission-reception conflict** (28): This constraint prevents that a node \( v \) can transmit a packet over link \( e \) while it is receiving data addressed to him over a different link \( f \), where \( f \in E^{in}(v) \) (Figure 124b). Assume node \( v \) starts transmitting over link \( e \) at time \( t_e \) and a transmission over link \( f \) starts at time \( t_f \). A conflict occurs whenever the transmission and reception of the two packets overlap in time. Then a conflict occurs whenever the transmission interval \([t_e, t_e + 1]\) and the reception interval \([t_f + d_f, t_f + d_f + 1]\) are not disjoint (Figure 124b). Moreover, if \( t_f + d_f > T \) we have to account for conflicts due to transmissions scheduled in previous frames. This corresponds to the constraint

\[ \forall v \in V, \forall e \in E^{in}(v), \forall t_e \in [0,T-1] \]

\[ X(e,t_e) \leq 1 \iff t_e < t_f + d_f + 1. \]
\[ \forall v \in V, \forall e \in E^m(v), f \in E^m(v), \forall t_e, t_f \in [0, T - 1] \forall k \in [0, L], \]
\[ X(e, t_e) + X(f, t_f) \leq 1 \iff t_f + d_f - kT < t_e < t_f + d_f - kT + 1 \] (16)

- **interference conflict (29)**: This constraint prevents that while a node \( v \) is receiving over an incoming link \( e \in E^m(v) \), correct reception is impaired due to a transmission on a different link \( f \) which is not an incoming link, i.e. \( f \not\in E^m(v) \) and \( e_{dst} \neq f_{dst} \) (Figure 124d). Assume a transmission over link \( e \) starts at time \( t_e \) and a transmissions over link \( f \) starts at time \( t_f \). Similarly to the concurrent reception conflict case, a conflict occurs whenever the reception of the two packets overlaps in time at \( v \) and \( f \in I(e) \). Let \( d_{f,e} = d(f_{src}, e_{dst}) \) denote the propagation delay (always measured in number of slots) from the source of link \( e \) to the destination of link \( f \). Then a conflict occurs whenever the two packet reception time intervals \([t_e + d_e, t_e + d_e + 1]\) and \([t_f + d_{f,e}, t_f + d_{f,e} + 1]\) are not disjoint. It is easy to verify that this corresponds to the constraint
\[ \forall v \in V, \forall e \in E^m(v), f \not\in E^m(v), \]
\[ t_e, t_f \in [0, T - 1], k \in [0, L] \text{ and } f \in I(e) \]
\[ X(e, t_e) + X(f, t_f) \leq 1 \iff t_e + d_e - 1 < t_f - kT + d_{f,e} < t_e + d_e + 1 \] (17)

where again the term \( kT \), accounts for conflicts between transmissions occurring in different frames.

We observe that this constraint captures only pairwise interference. For a more realistic interference characterization we need to consider the more general situation whereby a set of transmissions produces a non negligible interference at the receiver. This would require an exponential number of constraints which would not be feasible except for very small scenarios. Instead, we follow the approach proposed in [53] where a solution is computed at first considering only the constraints above. Then, ex-post, for the obtained schedule, we evaluate whether a SIR-violation occurs at the receiver. We have a SIR-violation at the destination of link \( e \), if:
\[ \frac{P \cdot \text{Gain}(e_{src}, e_{dst})}{\sum_{f \in E^m(v) : X(f, t_f) = 1} P \cdot \text{Gain}(f_{src}, e_{dst}) + N < \text{SIR}_{th}} \] (18)

Let \( A \) denote the set of links interfering with link \( e \) when they are all actives. If there is a SIR-violation at the destination of link \( e \), we augment the set of constraints by adding a new one ensuring that at most \(|A| - 1 \) of the links in \( A \) can co-exist when link \( e \) is active. After all the violations are accounted for with the new constraints (18) the problem is solved again. We proceed iteratively until no SIR-violations are observed.
(a) intra-frame reception-reception conflict

(b) intra-frame reception-reception conflict

(c) inter-frame reception-reception conflict

Fig. 125: concurrent reception conflicts
Another aspect that is necessary to consider in the formulation of our problem is data flow management. The following constraints describe the buffer dynamic of each sensor node.

- **Incoming messages** (30): Packets that are inserted into the buffer of node $v$ at time $t$ are packets either generated by node $v$, or completely received at time $t$. If node $v$ receives a packet at time $t$ over link $e$ with propagation delay $d_e$, then the source node of $e$ started to transmit data at time $t_e = t - d_e - 1$. If $t_e < 0$, the transmission started during a previous frame. Since traffic is assumed to be periodic, if link $e$ transmitted during the $k$-th previous frame at time $t_e$, it will transmit at time $t_e + kT$ during the current frame.

$$\forall v \in V, t \in [0, T - 1]$$
$$B_{In}(v, t) = g(v, t) + \sum_{k=0}^{i} \sum_{e \in E^{in}(v): 0 \leq t_e < T} X(e, t_e) \quad (19)$$

where $t_e = t + kT - d_e - 1$ and $d_e = d(e, src, e, dst)$.

- **Outgoing messages** (31): The total number of packets transmitted by node $v$ at time $t$ is the sum of the number of packets transmitted at time $t$ over the links departing from $v$.

$$\forall v \in V, t \in [0, T - 1]$$
$$B_{Out}(v, t) = \sum_{e \in E^{out}(v)} X(e, t) \quad (20)$$

- **Storage** (32): The number of packet inside the buffer during time $t$ are those that arrived at node $v$ before or during time $t$, minus those transmitted by $v$ before or during time $t$.

$$\forall v \in V, t \in [0, T - 1]$$
$$B_{Size}(v, t) = \sum_{r=0}^{t} B_{In}(v, r) + \sum_{r=0}^{t} B_{Out}(v, r) \quad (21)$$

- **Sink storage** (33): This constraint ensure that all generated traffic during a frame eventually reaches the sink.

$$B_{Size}(sink, T - 1) \geq \sum_{v \in V} \sum_{t \in [0, T - 1]} g(v, t) \quad (22)$$

The resulting optimization problem $OPT(T)$ takes the following form:
\[ \text{OPT}(T) : \min \sum_{e \in E} \sum_{t=0}^{T-1} X(e,t) \cdot T(x) \]  
subject to:
\[ \sum_{t=0}^{T-1} g(\text{sink},t) = 0 \]  
\[ \sum_{e \in E^{\text{out}}(\text{sink})} X(e,t) = 0 \]  
\[ \sum_{e \in E^{\text{out}}(v)} X(e,t) \leq 1 \]  
\[ \forall v \in V, \forall t \in [0,T-1] \]
\[ X(e,t_e) + X(f,t_f) \leq 1 \]  
\[ t_e + d_e - 1 < t_f + d_f - kT < t_e + d_e + 1 \forall e, f \in E^{\text{in}}(v), \]  
\[ \forall t_e, t_f \in [0,T-1], \forall k \in [0,L] \]
\[ X(e,t_e) + X(f,t_f) \leq 1 \]  
\[ t_f + d_f - kT < t_f < t_f + d_f + 1 - kT \]  
\[ \forall e \in E^{\text{out}}(v), f \in E^{\text{in}}(v), t_e, t_f \in [0,T-1], k \in [0,L] \]
\[ X(e,t_e) + X(f,t_f) \leq 1 \]  
\[ t_e + d_e - 1 < t_f - kT + d_f, f < t_e + d_e + 1 \text{ and } t_e + d_e < T \]  
\[ \forall e, f \in E, t_e, t_f \in [0,T-1], k \in [0,L] \text{ and } f \in I(e) \]
\[ B\text{In}(v,t) = g(v,t) + \sum_{k=0}^{L} \sum_{0 \leq t_e < T} X(e,t_e) \]  
\[ \forall v \in V, \forall t \in [0,T-1] \]
\[ B\text{Out}(v,t) = \sum_{e \in E^{\text{out}}(v)} X(e,t) \]  
\[ B\text{Size}(v,t) = \sum_{r=0}^{L} B\text{In}(v,r) + \sum_{r=0}^{L} B\text{Out}(v,r) \]
\[ \forall v \in V, t \in [0,T-1] \]
\[ B\text{Size}(\text{sink},T-1) \geq \sum_{v \in V} \sum_{t \in [0,T-1]} g(v,t) \]  
Given a solution of the problem \( \text{OPT}(T) \) we obtain the schedule \( S_t, t = 0, \ldots, N-1 \) as follows:
\[ S_t = \{ e \in E \mid X(e,t) = 1 \} \quad t = 0, \ldots, T-1 \]
9.4 Centralized heuristic

In this section we present a (class) of heuristics for the joint routing-scheduling problem. Given the complexity of the original problem, we propose a simple divide et impera scheme where we first determine packet routing and then, with the routing fixed, we schedule packets transmissions.

9.4.1 Routing/Conflict graph

For packets routing we can use any suitable routing algorithm, e.g., shortest path, geographic routing, etc. In addition, here we also propose a new routing scheme, which is based on a simple fluid flow model approximation of the network traffic, and is designed to account for interference among nodes transmissions, and which minimizes the same joint routing-scheduling objective function used in $\text{OPT}(T)$. For our solution, we will resort to well known optimization techniques used in modeling wireless networks which accounts for links conflicts/interference via the so called conflicts graph. We remark that this allows us to capture pairwise conflicts but not the more general interference patterns which arise in underwater networks. We found indeed that it is not possible to account for higher order interference using fluid traffic models. In the following, we first introduce our conflict graph and then derive the optimal routing.

Conflict Graph— We use a labeled conflict graph to capture conflict and interference constraints among different links. A conflict graph is a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices, each of them representing a link, and $\mathcal{E}$ is the set of edges, representing links conflicts, i.e., two links transmission conflict one another if and only if there is an edge between them. Conflict graphs has been widely used to model conflict/interference in wireless networks, where transmission conflicts occur when two links transmit at the same time. In underwater networks, because of the significant propagation delay, transmission of one link at a given time instant may conflict with another link transmission at a earlier or later time depending on the relative link propagation delays. To this end, we consider a labeled conflict graph where labels represent the relative delay which cause a conflict between links transmissions.

We say that a conflict relation $\text{Con}(e \rightarrow f)$, $e, f \in \mathcal{V}$, exists if transmission of link $e$ prevents reception at link $f$. This is represented in $G$ by the link $(e, f) \in \mathcal{E}$. We assign to the edge $(e, f)$ the label $c_{e,f}$, where $c_{e,f}$ is the conflict delay between $e$ and $f$, i.e., in other words, a link $e$ transmission at time $t + c_{e,f}$ conflicts with a transmission with a link $f$ transmission at time $t$.

As in the previous section, we need to consider different types of conflicts. First, because of the shared underwater medium, transmissions over different links can causes interference, i.e., when two links share the same receiver, or when when the sender of one link is within the interference range of the receiver. Second, we need to account for the physical limitation that one node can transmit over only one single link. We detail them below.
(Two links with a common node) We can have different type of conflicts:

1. \( e.\text{dst} = f.\text{dst} \) (Links \( e \) and \( f \) have the same destination): A node cannot receive two different packet at the same time. Hence we have \( \text{Con}(e \rightarrow f) \) and \( \text{Con}(f \rightarrow e) \) with conflict delay:
   \[ c_{ef} = -c_{fe} = d_f - d_e. \]

2. \( e.\text{src} = f.\text{src} \) (Links \( e \) and \( f \) have the same transmitter): A node cannot transmit two packet at the same time. Hence, we have \( \text{Con}(e \rightarrow f) \) and \( \text{Con}(f \rightarrow e) \) with conflict delay:
   \[ c_{ef} = c_{fe} = 0. \]

3. \( e.\text{src} = f.\text{dst} \) (Links \( e \) transmitter is link \( f \) receiver): A node cannot receive and transmit at the same time. We have \( \text{Con}(e \rightarrow f) \) and \( \text{Con}(f \rightarrow e) \) with conflict delay:
   \[ c_{ef} = -c_{fe} = d_f. \]

(Two links without a common node - Interference) In this case we consider whether a link reception is interfered by another link transmission. Observe that with a conflict graph we can only capture this type of interference, but not the more general scenario where a link reception is interfered by the superposition of a set of interfering links but is not interfered by the sole transmission of any single link. In the routing determination algorithm we will consider the conflict graph only - thus considering only pairwise interference. In the scheduling phase, nevertheless, we will account for the more general type of interference and schedule packet transmission accordingly.

In this setting, we have interference \( \text{Con}(e \rightarrow f) \) if the SIR ratio between a link \( f \) transmission received power by the \( f.\text{dst} \) and the interference generated by a link \( e \) transmission is below the SIR threshold \( \text{SIR}_{th} \), i.e., if

\[ \frac{P \cdot \text{Gain}(f.\text{src}, f.\text{dst})}{P \cdot \text{Gain}(e.\text{src}, f.\text{dst}) + N} < \text{SIR}_{th} \]

the corresponding conflict delay is again \( c_{ef} = d_f - d_e. \)

### 9.4.2 Routing

We determine the routing by solving a suitable optimization problem. For the sake of simplicity, we consider a simple fluid flow model of the network and determine the routing which minimize our joint routing-scheduling objective function. The idea is that if we want to decouple routing from scheduling, we must do away with the low level packet transmission details and look at the network traffic at a coarse time scale. This can be conveniently done by considering a fluid model of network traffic which only consider the aggregate/average behavior across links.
Traffic Model— In this setting, network traffic consists of a set of flows $H$, with each flow $h \in H$ characterized by a source node $src(h)$, a destination node $dst(h)$ and a flow rate $x_h$. We will measure flow rate in number of packets per frame, i.e. $x_h = 1$ means that flow $h$ need to send one packet per frame. Flow traffic is (possibly) routed by the network along one or multiple paths from source to destination. We denote by $s_{he}$ the rate of flow $h$ on link $e$. The flow conservation law implies that for $h \in H$, $u \in U$:

$$x_h + \sum_{l \in E^{in}(u)} s_{hl} = \sum_{l \in E^{out}(u)} s_{hl} \quad u = src(h), h \in H$$ (35)

$$\sum_{l \in E^{in}(u)} s_{hl} = \sum_{l \in E^{out}(u)} s_{hl} \quad u \neq src(h), dst(h), h \in H$$ (36)

$$\sum_{h \in H} s_{hl} = c_l \quad l \in E$$ (37)

where the first two equations simply states that for any node traversed by the flow $h$, except for the destination node $dst(h)$, the flow entering the node must be equal to the flow exiting it. The last equation states that the link $l$ transmission rate $c_l$ is equal to the sum of all the flow transmission rate traversing it.

Conflict Constraints— To account for transmission conflicts among the different links we use the conflict graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ previously computed.

Consider the maximal cliques in $\mathcal{G}$. A maximal clique $Q = (V_Q, E_Q)$ in $\mathcal{G}$ is a maximal complete sub-graph of $\mathcal{G}$. A maximal clique in the link contention graph denotes a distinct contention region because any pair of links $(u_1, v_1), (u_2, v_2) \in V_Q$ can interfere with one another. Hence each maximal clique represents a distinct resource with the upstream nodes of the links in $Q$ share.

In this model, only links that are in different cliques can transmit without conflicting with other links. As a result, cliques determine constraints on the links transmission rate. Since links within the same clique cannot share the transmission rate we obtain the following constraints

$$\sum_{l \in V_Q} c_l \leq t \quad Q \in \mathcal{Q}.$$ (38)

where $\mathcal{Q}$ denote the set of maximal cliques. Here, the variable $t$ represents the time required to transmit all traffic from source to destination. Clearly $t$ must be larger than the sum of the rate over each clique (intuitively, since transmission of any pair of links in the same clique conflict with one another, transmission within a clique must be serialized; the overall transmission time $t$ must be larger than any of these sums).

Problem Formulation—

We compute the optimal routing by first solving the following optimization problem $\text{Flow}_\text{OPT}$:
 Deliverable D4.2

\[ \text{Flow\_OPT: min } \alpha t + \sum_{h \in H} c_h \]
\[ \text{s.t. } x_h + \sum_{l \in E_{\text{in}}(u)} s_{hl} \leq \sum_{l \in E_{\text{out}}(u)} s_{hl}, \quad u = \text{src}(h), h \in H \quad (39) \]
\[ \sum_{l \in E_{\text{in}}(u)} s_{hl} \leq \sum_{l \in E_{\text{out}}(u)} s_{hl}, \quad u \neq \text{src}(h), \text{dst}(h), h \in H \quad (40) \]
\[ \sum_{l \in E_{\text{in}}(u)} s_{hl} \leq c_l, \quad l \in E \quad (41) \]
\[ \sum_{l \in E_{\text{in}}} c_l \leq t, \quad Q \in \mathcal{Q} \quad (42) \]

with variables \( s_{hl}, h \in H, l \in E, c_l, l \in E \) and \( t \). The constraints are the flow conservation constraints (39)-(40), the link load constraints (41) and the the conflict clique constraints (42). The objective function is a weighted sum of the total transmission time \( t \) and the energy spent by all nodes: with \( \alpha = 0 \) the objective is to minimize the energy; with \( \alpha \to \infty \) the objective is to minimize the transmission time.

We must observe that the problem Flow\_OPT is deceivingly simple: Flow\_OPT is a LP problem which can be efficiently computed via standard techniques; but at a closer look we must observe that finding the set of maximal cliques \( \mathcal{Q} \) is a well known NP-HARD problem and the number of maximal cliques is exponential in the number of the nodes of the conflict graphs. Hence, only for small instances the computation of \( \mathcal{Q} \) can be carried out via known algorithms, i.e., the Bron-Kerbosch algorithm. For larger instances, we need to resort to heuristics. In our implementations, we simply replace \( \mathcal{Q} \) with a set \( \mathcal{Q}' \) of cliques which we compute via clique computation heuristics.

Routing Computation—

Problem Flow\_OPT is a LP problem. Its solution includes the rates \( s_{hl}, h \in H, l \in E \) of each (fluid) flow along each link. Implicitly, this solution specifies a multipath routing across the network which identifies the (possible multiple) paths along which traffic traverse the network from source to destination. Since we are interested in a simple single-path routing for each flow, we determine the flow routing by simply rounding \( s \) as follows (see Fig. 126). For each flow \( h \in H \) we initialize the path \( \pi_h \) with source node \( \text{src}(h) \). Then, until we reach the destination \( \text{dst}(h) \), we proceed hop-by-hop traversing at each step the link which carries the largest fraction of flow \( h \) traffic from the current node (step 8).

9.4.3 Scheduling

Our scheduling heuristic is inspired by recent work by [51] which proposes a simple greedy heuristic for scheduling in a UWSN. In [51], following the observation that the scheduling problem can be regarded as a coloring problem of the conflict graph, where a color assigned to a node corresponds to a transmission slot in a frame, the authors presents a vertex-coloring greedy heuristic, which accounts for the interference delay expressed by the labels associated to the conflict graph links, which
schedules links according to their loads (and breaking ties in favor of links with higher number of conflicts in the network).

Our proposed solution extends the work in [51] in several directions. First, we consider a full interference model. We do not consider only the conflict graph, but, while constructing the schedule, we keep track of the interference generated by already scheduled nodes at all receivers in the networks. When scheduling a new node we choose available slots that avoid both conflicts between links as modeled by the conflict graph and prevents receivers interference due to the transmission by all already scheduled transmissions. Second, our solution accounts for a periodic schedule by explicitly considering the interference among transmissions occurring in different frames. Finally, we consider more metrics than those considered in [51]. While their work consider only load as metric to rank links, we consider and propose alternative metrics, e.g., number of conflicting links or the ratio between the link load and available slots. In practice, given the short execution time required by the heuristic, we can compute the schedule using different metrics for the ranking and just take the best solution.

The pseudo-code of the scheduling algorithm is shown in Fig. 127. The algorithm takes in input the conflict graph $G$, the link load $L(e), e \in \mathcal{E}$, measured as the number of packets to be transmitted by $e$ per frame, and the frame length $T$ and returns the matrix schedule $S$. The row $S[l], l \in E$ is link $l$ schedule: $S[l][t] = TRUE$ means that link $l$ is scheduled to transmit a time slot $t$ (or, put it in another way, node $e.$src is scheduled to transmit to node $t(v)$ at time slot $t$).

As a preliminary step we first determine the link load $L(e), e \in E$. This is computed by considering the number of packets generated by each node $v \in V$ per frame $\sum_{t=0,...,T-1} g(v,t)$ and the routing $\pi_h, h \in H$. The algorithm then uses a simple greedy policy to schedule links transmissions. In the code, $E'$ is the set of links yet to be scheduled which is initialized with the set of the links to be scheduled, i.e., the set of links with positive load (line 2). Then, until all links transmissions are scheduled,
the algorithm iteratively sorts $E'$ (the metrics for link ranking are discussed below) (line 6), picks the first link, $e_1$, and scheduled it (line 7) by determining the first $L(e_1)$ available slots. It then updated the schedule $S$ (line 9) and the data structures which keep track of links conflicts (lines 10-12) and interference (lines 13-15).

The algorithm uses two matrices, $M$ and $P$, to keep track of the links conflicts and interference due to other node transmissions at receivers nodes, respectively. For each link $e \in E$, $M[e]$ is a Boolean array which shows which are the unavailable time slot for transmissions. $M[e][r] = TRUE$ means that due to conflicts with already scheduled links, link $e$ cannot transmit at time slot $t$. For each receiver $v \in V$, $P[v]$ is a real valued array which keeps track of the interference generated by the transmission of the already scheduled links at receiver node $v$. Whenever a new link is scheduled, the two matrices are updated by adding which new links/slots are unavailable due to $e_1$ transmissions, and interference its transmissions generated at other receivers.

The function $Available$ (Fig. 128) simply determines the first $k$ available slot for a given link $e$. To this end, starting from the first slot in the frame, it simply scans the matrices $P$ and $M$ to determine whether transmissions in slot $j$ would suffer interference from the transmission of the already scheduled links.
Algorithm 3: Function Available

1. Function Available(e, M, P, k);
   input : link e; number of packets to transmit k
   output Node Schedule sched
2. begin
3. \( j \leftarrow 1; \)
4. forall \( 1 \leq i \leq k \)
5. while \( \frac{P \cdot \text{Gain}(u, \text{src}, u, \text{dst})}{P[u, \text{dst}, \text{Mod}(j + d_e, T)]} \leq \text{SIR}_{th} \) OR \( M[e][j] \)
6. \( j \leftarrow j + 1; \)
7. if \( j > T \) then
8. it is possible to schedule e within T slots;
9. return Error;
10. end
11. sched[i] \( \leftarrow j; \)
12. \( j \leftarrow j + 1; \)
13. end
14. return sched;
15. end

Fig. 128: Function Available

\( \frac{P \cdot \text{Gain}(u, \text{src}, u, \text{dst})}{P[u, \text{dist}, \text{Mod}(j + d_e, T)]} \leq \text{SIR}_{th} \) or would otherwise conflict with another link transmission \( M[e][j] = \text{TRUE} \) (lines 5-7), otherwise it schedules a transmission (line 8).

We observe that similarly to the optimal ILP model we need to choose the frame length \( T \) before executing the scheduling algorithm (observe also that for small enough frame length the heuristic might not able to find a feasible solution). To this end, we follow the same approach taken for the optimal ILP model and use binary search to identify the optimal frame length \( T^* \).

Ranking Metrics

The performance of the scheduling heuristic is heavily affected by the ranking metric which ultimately determines the order according to which links are scheduled for transmission. The original work in [51] considered only a simple load metric \( L(e) \). We investigated different alternative metrics which account for the additional following quantities:

- \( \text{conf}(e) \), number of links which conflict with link \( e \) transmission. For a given link \( e \) we consider the number of possible conflict generated by link \( e \) transmissions, i.e., the number of slots which would not be available to other links for transmission if link \( e \) is scheduled to transmit a packet. As a variation we can consider
only the number of not yet scheduled links which would conflict with \( e \). We will denote such number with \( \text{conf}^f(e) \):

- \( \text{free}(e) \), the number of still available slots for a link \( e \) transmission in the current frame. Observe that this quantity must be recomputed every time a new link is scheduled since its transmissions could conflicts with some of the previously free slots.

We also considered the following metrics:

- \( \text{LOAD} \) metric = \( L(e) \) This is the metric proposed in [51]. Priority is given to links with more traffic to send. Ties are broken in favor of links with more conflicts;
- \( \text{LpFS} \) metric = \( \frac{L(e)}{\text{free}(e)} \). Priority is given to the links with higher number of packets to transmit over free slot ratio. The idea is that priority should be given to links with less number of available free slots per packet;
- \( \text{LpFSC} \) metric = \( \frac{L(e)}{\text{free}(e) \cdot \text{conf}(e)} \). Here priority is given to links with higher number of packets to transmit over the product of free slots and number of conflicts per transmitted packets. Here the idea is to favor links which has less free slots available and that cause less conflicts with other links per packet transmitted;
- \( \text{LpFSC} \) metric = \( \frac{L(e)}{\text{free}(e) \cdot \text{conf}^f(e)} \). As above but conflicts are considered only with respect to link not already scheduled. The rationale is that conflicts with links already scheduled are not relevant and should not be accounted for.

The idea is that the scheduling order should also account for the number of conflicts (and hence a reduced number of available slots for the other links) caused by a link transmissions and/or the number of still available slot in the current frame. We also explored other combinations/metrics which exhibited worse overall performance. They have been thus omitted.

### 9.5 Performance Evaluation

We have conducted a thorough set of experiments to evaluate the performance of the joint optimal routing/scheduling solution and of the heuristic. We implemented the ILP formulation of the optimization model in CPLEX [54] solver, and the heuristics in MATLAB. In this Section we present the results based on the 20 and 40 nodes topologies described in Section 4 where we have sensor nodes in communication with a sink node in a multi-hop fashion. The propagation speed is assumed to be 1500 m/s, while the data bit rate is 1000 bps. We have first investigated a packet size of 200 bits, which results in a slot boundary of 0.2s, as by assumption nodes can transmit a packet during a slot, considering the minimum frame size and end-to-end latency in number of slots for both the optimal model and the heuristic. We have then considered different packet sizes, resulting in different slot durations, to investigate the minimum frame size obtained by the proposed solutions when varying the ratio between transmission delay and propagation delay. Table 3 shows the summary of our simulation parameters and traffic rates. In our analysis we focus on the following...
metrics: 1) minimum frame size $T$ needed to transmit all generated packets, 2) end-to-end packet latency defined as number of slot it take, on average, for a packet to be delivered to the sink.

We have performed experiments for increasing level of network traffic: 25%, 50%, 75% and 100%. In all the experiments we assume a node generates (at most) one packet per frame, but only a (randomly selected) fraction of the nodes generate traffic. 1/4 in the 25% scenario; 1/2 in the 50% and so on.

For every setup we perform experiments considering pairwise/binary and complete interference model. In Tables 4 and 5 and 6, 7 we show the results obtained assuming a slot duration of 0.2s for a topology of 20 nodes, considering both binary and complete interference models. The results for a 40 nodes topology are presented in Tables 8, 9, and 10, 11. For the heuristics, we show the best results obtained with the different routing/ranking schemes considered. Given the low computationally complexity of the heuristic we found that in practice it is reasonable to just run them all and take the best result.

The results show that the heuristic yields close-to-optimal results for both topologies and for all load levels. On average the difference between optimal value for minimum frame size and heuristic estimation is just 1.625 slots, with the heuristics running at a fraction of the cost, often within few seconds. This seem to be promising for a solution which aims to scale to large networks topologies.

We have performed experiments for increasing level of network traffic: 25%, 50%, 75% and 100%. In all the experiments we assume a node generates (at most) one packet per frame, but only a (randomly selected) fraction of the nodes generate traffic. 1/4 in the 25% scenario; 1/2 in the 50% and so on.

For every setup we perform experiments considering pairwise/binary and complete interference model. In Tables 4 and 5 and 6, 7 we show the results obtained assuming a slot duration of 0.2s for a topology of 20 nodes, considering both binary and complete interference models. The results for a 40 nodes topology are presented in Tables 8, 9, and 10, 11. For the heuristics, we show the best results obtained with the different routing/ranking schemes considered. Given the low computationally complexity of the heuristic we found that in practice it is reasonable to just run them all and take the best result.

The results show that the heuristic yields close-to-optimal results for both topologies and for all load levels. On average the difference between optimal value for minimum frame size and heuristic estimation is just 1.625 slots, with the heuristics running at a fraction of the cost, often within few seconds. This seem to be promising for a solution which aims to scale to large networks topologies.

Comparing the results for the different interference models, we observe that for the 20 nodes topology we obtain the same results for both the binary and the complete interference model in most of the cases, while for the 40 nodes topology we obtain longer frame durations, i.e., lower throughput, when we consider the complete interference model. This suggests that in general using the simple binary interference model, as widely adopted in the literature, is just an approximation. Our experiments show that pairwise interference can be acceptable in small scenarios, where each single node transmission is able to interfere destructively with any other transmissions. In such scenarios considering higher order interference does not provide any benefit to the solution accuracy. In our experiments this occurred in the 20 nodes topology scenario where we obtained the same results under both the interference models. On the other hand, in larger scenarios, with nodes further away from

---

Table 3: Simulation parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data rate</td>
<td>1000bps</td>
</tr>
<tr>
<td>Propagation speed</td>
<td>1500m/s</td>
</tr>
<tr>
<td>Packet size</td>
<td>100bits, 200bits, 500bits and 1000bits</td>
</tr>
<tr>
<td>Slot boundary</td>
<td>0.1s, 0.2s, 0.5s and 1s</td>
</tr>
<tr>
<td>Topology size</td>
<td>20 nodes, 40 nodes</td>
</tr>
<tr>
<td>SNR</td>
<td>9dB</td>
</tr>
<tr>
<td>SIR</td>
<td>6.5dB</td>
</tr>
</tbody>
</table>
one another, and where the interference from a single far away node might not be sufficient to disrupt communication, the superposition of multiple distant nodes can still result in enough high interference to prevent packet reception. These phenomena cannot be captured using the simplified binary interference model and, as our results show, would provide optimistic interference evaluation and optimistic, but erroneous, results.

Table 4: Minimum frame size (slots)
20 nodes - Binary interference - (0.2s)

<table>
<thead>
<tr>
<th>Traffic rate</th>
<th>Optimum</th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>25%</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>50%</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>75%</td>
<td>16</td>
<td>19</td>
</tr>
<tr>
<td>100%</td>
<td>22</td>
<td>24</td>
</tr>
</tbody>
</table>

Table 5: End-to-end latency (slots)
20 nodes - Binary interference - (0.2s)

<table>
<thead>
<tr>
<th>Traffic rate</th>
<th>Optimum</th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>25%</td>
<td>6.666</td>
<td>6.66</td>
</tr>
<tr>
<td>50%</td>
<td>11.12</td>
<td>13.19</td>
</tr>
<tr>
<td>75%</td>
<td>12.98</td>
<td>16.76</td>
</tr>
<tr>
<td>100%</td>
<td>15.94</td>
<td>23.75</td>
</tr>
</tbody>
</table>

Table 6: Minimum frame size (slots)
20 nodes - Complete interference - (0.2s)

<table>
<thead>
<tr>
<th>Traffic rate</th>
<th>Optimum</th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>25%</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>50%</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>75%</td>
<td>16</td>
<td>19</td>
</tr>
<tr>
<td>100%</td>
<td>22</td>
<td>23</td>
</tr>
</tbody>
</table>

Table 7: End-to-end latency (slots)
20 nodes - Complete interference - (0.2s)

<table>
<thead>
<tr>
<th>Traffic rate</th>
<th>Optimum</th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>25%</td>
<td>6.666</td>
<td>6.66</td>
</tr>
<tr>
<td>50%</td>
<td>11.12</td>
<td>13.19</td>
</tr>
<tr>
<td>75%</td>
<td>12.98</td>
<td>16.76</td>
</tr>
<tr>
<td>100%</td>
<td>15.94</td>
<td>23.75</td>
</tr>
</tbody>
</table>

Table 8: Minimum frame size (slots)
40 nodes - Binary interference - (0.2s)

<table>
<thead>
<tr>
<th>Traffic rate</th>
<th>Optimum</th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>25%</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>50%</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>75%</td>
<td>33</td>
<td>34</td>
</tr>
<tr>
<td>100%</td>
<td>44</td>
<td>45</td>
</tr>
</tbody>
</table>

Table 9: End-to-end latency (slots)
40 nodes - Binary interference - (0.2s)

<table>
<thead>
<tr>
<th>Traffic rate</th>
<th>Optimum</th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>25%</td>
<td>19.6</td>
<td>20.22</td>
</tr>
<tr>
<td>50%</td>
<td>18.01</td>
<td>29.34</td>
</tr>
<tr>
<td>75%</td>
<td>20.14</td>
<td>39.95</td>
</tr>
<tr>
<td>100%</td>
<td>44.37</td>
<td>51.37</td>
</tr>
</tbody>
</table>

In Tables 12 and 13 we show the results obtained assuming different slot durations (0.1s, 0.5s and 1s) for 20 nodes topology, considering binary and complete interference model respectively. The results for a 40 nodes topology are presented in Tables 14 and 15. In general we can see that when the transmission delay increases the number of slots per frame increases as well. When the transmission delay is comparable or longer than the propagation delay is more difficult to schedule transmissions which can overlap in time with no collision at the receiver, thus increasing the channel utilization. We can also observe that, when the slot duration increases,
Table 10: Minimum frame size (slots) 40 nodes - Complete interference - (0.2s)

<table>
<thead>
<tr>
<th>Traffic rate</th>
<th>Optimum</th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>25%</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>50%</td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>75%</td>
<td>37</td>
<td>39</td>
</tr>
<tr>
<td>100%</td>
<td>48</td>
<td>48</td>
</tr>
</tbody>
</table>

Table 11: End-to-end latency (slots) 40 nodes - Complete interference - (0.2s)

<table>
<thead>
<tr>
<th>Traffic rate</th>
<th>Optimum</th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>25%</td>
<td>19.6</td>
<td>25.02</td>
</tr>
<tr>
<td>50%</td>
<td>20.15</td>
<td>30.89</td>
</tr>
<tr>
<td>75%</td>
<td>32.27</td>
<td>39.44</td>
</tr>
<tr>
<td>100%</td>
<td>59.70</td>
<td>61.85</td>
</tr>
</tbody>
</table>

considering only the binary interference model provides erroneous results even for the 20 nodes topology case, while for shorter slot durations binary and complete interference models result in the same frame length. This suggests future investigation considering more topologies.

We can see that in general increasing the slot duration, the gap between the optimum and the heuristic frame length solution increases as well. In fact, when the ratio between transmission and propagation delay is such that finding a schedule that optimize the channel usage is more difficult, the more simple approach used by the heuristic results in a larger gap with respect to the optimal solution. However, this gap is on average shorter than 4.25 slots which is definitely acceptable considering the lower computational time required by the heuristic: In these set of experiments, the heuristic always run in less than two minutes corresponding to up to 4 orders of magnitude reduction in computational time with respect to the ILP solver.

Table 12: Minimum frame size (slots) 20 nodes - Binary interference

<table>
<thead>
<tr>
<th>Traffic rate</th>
<th>0.1s</th>
<th>0.5s</th>
<th>1.0s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Optimum</td>
<td>Heuristic</td>
<td>Optimum</td>
</tr>
<tr>
<td>25%</td>
<td>6</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>50%</td>
<td>11</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>75%</td>
<td>16</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>100%</td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
</tbody>
</table>

Table 13: Minimum frame size (slots) 20 nodes - Complete interference

<table>
<thead>
<tr>
<th>Traffic rate</th>
<th>0.1s</th>
<th>0.5s</th>
<th>1.0s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Optimum</td>
<td>Heuristic</td>
<td>Optimum</td>
</tr>
<tr>
<td>25%</td>
<td>6</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>50%</td>
<td>11</td>
<td>14</td>
<td>11</td>
</tr>
<tr>
<td>75%</td>
<td>16</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>100%</td>
<td>22</td>
<td>24</td>
<td>22</td>
</tr>
</tbody>
</table>
Table 14: Minimum frame size(slots) 40 nodes - Binary interference

<table>
<thead>
<tr>
<th>Traffic rate</th>
<th>0.1s</th>
<th>0.5s</th>
<th>1.0s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Optimum</td>
<td>Heuristic</td>
<td>Optimum</td>
</tr>
<tr>
<td>25%</td>
<td>12</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>50%</td>
<td>21</td>
<td>24</td>
<td>21</td>
</tr>
<tr>
<td>75%</td>
<td>33</td>
<td>37</td>
<td>33</td>
</tr>
<tr>
<td>100%</td>
<td>44</td>
<td>46</td>
<td>44</td>
</tr>
</tbody>
</table>

Table 15: Minimum frame size(slots) 40 nodes - Complete interference

<table>
<thead>
<tr>
<th>Traffic rate</th>
<th>0.1s</th>
<th>0.5s</th>
<th>1.0s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Optimum</td>
<td>Heuristic</td>
<td>Optimum</td>
</tr>
<tr>
<td>25%</td>
<td>12</td>
<td>14</td>
<td>12</td>
</tr>
<tr>
<td>50%</td>
<td>24</td>
<td>27</td>
<td>24</td>
</tr>
<tr>
<td>75%</td>
<td>37</td>
<td>44</td>
<td>37</td>
</tr>
<tr>
<td>100%</td>
<td>48</td>
<td>50</td>
<td>48</td>
</tr>
</tbody>
</table>

Table 16: 20 nodes topology - Complete interference model

<table>
<thead>
<tr>
<th>Traffic rate</th>
<th>Load</th>
<th>LpFSC</th>
<th>LpFSC′</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25%</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>33</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>75%</td>
<td>48</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>58</td>
<td>48</td>
</tr>
</tbody>
</table>

Table 17: 40 nodes topology - Complete interference model

<table>
<thead>
<tr>
<th>Traffic rate</th>
<th>Load</th>
<th>LpFSC</th>
<th>LpFSC′</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25%</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>33</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>75%</td>
<td>48</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>58</td>
<td>48</td>
</tr>
</tbody>
</table>

We now turn our attention on the performance of the heuristic under different routing strategies and under the different link ranking metrics. We compared the shortest path (all links with unitary cost) and flow optimal routing algorithms and in both cases we computed the schedule according to the four proposed link metrics. In Table 16 and 17 we summarize the results for the 20 and 40 nodes topologies where slot duration is equal 0.2s, respectively. For the sake of simplicity we only
report the results for the complete interference model in the case where packet size is assumed to be 0.2s.

From the tables we can observe that in these scenarios shortest path and fluid optimal model yield very close results. We believe that this largely due to the fact that since our problem objective function is the network overall transmission cost, given that transmission costs are all equal, this is equivalent to minimize the number of transmissions which is indeed what shortest path routing with unitary cost does. On the other hand, we must observe that the fluid optimal model requires the solution of a LP and requires the computation of all maximal cliques (which we only computed using an ad-hoc heuristic for keeping the computational complexity low), while shortest path routing is easily computed using the Dijkstra Algorithm which is very efficient. Comparing the different links ranking metrics we see there is no clear winner. In general Load which is the basic ranking scheme proposed in [51] exhibit the worst performance overestimating the schedule length significantly on several problem instances. In general, the ranking metrics we proposed which combines load, number of free slot and conflicts generated yield good results. Given the low computational complexity of the scheduling algorithm it is reasonable to just run them all and take the one with the best result.

10 On the impact of different scheduling and retransmission techniques on hierarchical aggregation-aware routing in underwater networks

So far, we have shown that the delivery of data to the sink throughout multihop underwater networks is possible, but may be subject to very high delays and loss of packets in case the network is congested. For example, this is the case for the SUN protocol. One way to reduce congestion is to aggregate data packets on the way to the sink, so that the sink gathers a compressed view of the quantity being sensed. In a real scenario, this may be the baseline mode of operations. The communications could be temporarily re-configured to employ non-aggregated transmissions in case the sink requires higher quality data, but may revert to aggregated transmissions after some time. The purpose of this section is to show aggregated data transmissions provide good performance in multihop underwater networks, and are thus a promising working direction, both inside and outside of CLAM.

To do this, we consider a hierarchically organized converge-casting underwater network, where all data have to be transmitted to a sink. The hierarchy is created by setting up clusters where one node is the cluster-head. The cluster creation process is automatically carried out by the Energy-efficient aDaptive hiErarchical and robustT Architecture (EDETA) protocol. Once the clusters are created, the cluster-heads form a tree structure, and the packets routed towards the sink always travel the hierarchy upstream, from the generating nodes to its cluster-head, then to other cluster-heads, until the sink is finally reached. Whenever the children nodes in a
cluster send packets to a cluster-head, we assume that the latter aggregates the contents of the packets, and sends only one packet to its own cluster-head.

The channel access protocol to be employed to administer transmissions within a cluster, as well as among different clusters, is a design choice. In the following, the impact of different channel access techniques, with and without ARQ, will be considered and evaluated.

The rest of this Section is organized as follows. The general description of EDETA is provided in Section 10.1 the channel access protocols employed along with EDETA are described in Section 10.2. Section 10.3 details the results of the performance evaluation, and Section 10.4 draws some concluding remark.

10.1 The EDETA protocol

EDETA (Energy-efficient aDaptive hiErarchical and robuST Architecture) is a routing protocol originally proposed for WSNs [55] and recently adapted to UWSNs [56]. EDETA is a hierarchical protocol: the nodes arrange themselves in clusters, where one of them acts as the cluster-head (CH). The CHs form a tree structure in order to send the collected and aggregated data from the other nodes to the sink in a multihop manner.

EDETA operations are arranged in two separate phases: (i) the initialization phase and (ii) the normal operation phase. During the initialization phase, the clusters are created and the CHs are elected. During the normal operation phase, the nodes send their data periodically to their CHs. This takes place at scheduled time epochs, in accordance with the chosen channel access policy (see also Section 10.2). Finally, cluster-heads send their data to their parents until the data reaches the sink.

An enhanced version of EDETA, called EDETA-e (for EDETA-enhanced), also allows the designers of the network to plan which nodes will be CHs. In this variant of the protocol the initialization phase need be carried out only once.

The protocol also defines a scheduling mechanism in order to avoid packet collisions during the normal operation phase. The schedule originally proposed for this phase is TDMA which, however, may be inefficient [57] due to the large propagation delays that are typical of UWSN, and that would require to include large guard times in the slots of the TDMA schedule. We therefore choose different scheduling techniques, which are detailed in the following sections.

10.2 Scheduling and Retransmission techniques

As previously stated, UWSNs suffer from high propagation delays, which can make the traditional TDMA and CSMA medium access techniques inefficient [58]. Scheduling the transmissions allows to avoid collisions and can reduce the propagation delay, taking advantage of it and overlapping them.
However, it is also necessary to introduce an extra time in the schedule to allow the retransmission of the packets in case a packet error occurs.

To evaluate the impact of different scheduling and retransmission techniques on EDETA, we will consider two different scheduling techniques and two different retransmission techniques.

In what follows, we introduce different combinations of the scheduling and retransmission techniques used to replace the original scheduling and retransmission technique of EDETA. The delay-aware schedule has been implemented using the simplified set of schedule constraints proposed by van Kleunen et al. in [59].

T Ack

T Ack is a TDMA schedule with acknowledgement in case of data packet loss. A TDMA schedule is used by the nodes to send their data and an ACK is sent back when the data is correctly received. Since this schedule is not delay-aware, the slots are extended to include a time period equal to the maximum propagation time in the network. Each transmission is scheduled twice, in order to provide a backup slot in case a data packet error occurs.

T noAck

T noAck is a TDMA schedule without acknowledgement. A TDMA schedule is used by the nodes to send their data but no ACK is sent back. Hence, the TDMA slot will last just the time needed for the packet transmission and the signal to propagate to the maximum distance.

D Ack

D Ack is a delay-aware schedule with acknowledgements in case of data packet loss. A delay-aware schedule is used by the nodes to send their data and an ACK is sent back when the data is correctly received. Each transmission is scheduled twice, so as to provide a backup slot in case of a data packet error occurs.

D noAck

D noAck is a delay-aware schedule without acknowledgement. A delay-aware schedule is used by the nodes to send their data but no ACK is sent back. Each packet is scheduled to arrive at the destination right after the previous one in the schedule.
DnoAck2

DnoAck2 is a delay-aware schedule without acknowledgement. A delay-aware schedule is used by the nodes to send their data but no ACK is sent back. As in the previous section, each packet is scheduled to arrive at the destination right after the previous one in the schedule, but is scheduled and transmitted twice.

10.3 Performance evaluation

We proceed to the evaluation of the EDETA-e protocol along with the scheduling policies described in the previous section. For this evaluation, we focus on scenario 3 (see Section 4 for reference).

The configuration phase of the EDETA-e protocol was set to last for 100 minutes. After that, the nodes start transmitting packets (provided that they have any in queue) according to the rules of the transmission policies. These packets have payloads of 100, 500 or 1000 bytes, as described in Section 4. After that, each CH aggregates these data packets and sends one packet (of the same size of each incoming packet) to its next CH in the tree structure, until the sink is reached.
10.3.1 TnoAck vs. DnoAck

In these experiments, the performance of a TDMA schedule and a delay-aware schedule, both of them without acknowledgement, were evaluated.

Given the hierarchical nature of the protocol, we note that the packet delivery delay is heavily influenced by the network topology and that the propagation delay is not the dominant factor because the delay caused by the transmissions through the CH is much higher. As it can be seen from Figure 129a, as the packet length increases, the delay increases because the CH employs more time to collect all packets from its children nodes.

Also, as depicted in Figure 129b, the energy consumption of the leaf nodes is the same for the two cases since, in both alternatives, the nodes only send their data without waiting for any acknowledgement. On the other hand, Figure 129c shows that CHs consume less energy when they use the delay-aware schedule compared with when they use the TDMA schedule. This difference in energy consumption is due to the initialization phase which is slightly different for TnoAck and DnoAck.
10.3.2 TAck vs. DAck

In these experiments, the performance of a TDMA schedule and a delay-aware schedule, both of them with acknowledgement and a scheduled backup transmission are evaluated. Due to this backup transmission scheduled right after the main transmission, the packet loss rate is 1% in every scenario and the duplicate data packet rate is around 10%.

Focusing on the delays, when acknowledgements and poll messages are introduced, the number of TDMA slots increase. Nevertheless, the delay-aware schedule can optimize the data transmissions and by doing so it can reduce the delays as depicted in Figure 130a.

Figures 130b and 130c show the energy spent per data byte sent by the leaf nodes and the CHs respectively. One can see that the difference between TAck and DAck in terms of the energy required to send data becomes lower as the data packet length increases. This is due to the overhead introduced by the polling signaling used by the TDMA schedule, which is not used by the delay-aware schedule.

![Figure 130a: Delay](image)

![Figure 130b: Leaf nodes energy consumption](image)

![Figure 130c: CH nodes energy consumption](image)

Fig. 131: DAck and vs DnoAck2 comparison
10.3.3 DAck vs DnoAck2

In this section, we compare the delay-aware schedule with acknowledgement (DAck) against the delay-aware schedule with two scheduled transmissions and no acknowledgement (DnoAck2).

In terms of packet loss, duplicate packets and packet delay, the DnoAck2 alternative behaves as expected. The packet loss ratio is the same as the DAck alternative since in both cases the data will be lost if both scheduled data transmissions fail. However, DnoAck2 has 100% duplicate data since each packet is sent twice. On the other hand, as shown in 131a the delay of the DnoAck2 alternative is the smallest one of the two alternatives.

In terms of energy consumption, when sending packets of 100 bytes, the DnoAck2 alternative is more energy efficient for the leaf nodes. But, in the other alternatives, the DAck is remarkably more efficient. This is due to the overhead of transmitting the packet twice (DnoAck2) or just sending back one ACK (DAck).

10.4 Conclusions

In this section, different scheduling and retransmission techniques applied to the hierarchical routing protocol EDETA-e have been simulated. Their performance has been analyzed in terms of energy consumption, delivery delay, packet loss rate and duplicate packets.

Results have shown that, given the relatively small transmission range of the networks simulated, there is no difference in delay and energy consumption during the normal operation phase of the TnoAck and DnoAck alternatives. However, when the rest of the control packets of the EDETA protocol are introduced (ACK and POLL) the packet delay in the TAck alternative is higher than in the DAck alternative. This difference in the delay also comes with a high energy consumption of the CHs.

The DAck and TAck alternatives also have a retransmission technique based on scheduling a backup transmission right after the primary transmission. Since both of them use the same technique, the loss rate and duplicate packet rate is the same in both alternatives. If the packet error rate can be known a priori or can be estimated, the nodes can dynamically adjust the number of backup transmissions in order to achieve a desired packet lost rate.

Finally, the DnoAck2 alternative offers some interesting results. Since there is no acknowledgement packet sent back, the CHs are able to save energy; however, this saved energy is used by the leaf nodes to perform the retransmissions. As for the other MAC protocols tested in this deliverable, the protocols can save significant energy by avoiding unnecessary receptions. If this can be done, one can choose, e.g., the policy that provides the lowest packet loss rate, or the lowest overhead, depending on the simulation scenario.

The costs of hierarchy creation and maintenance, of the setup of a shared transmission schedule among the nodes, and of the distribution of a common time refer-
ence are still to be evaluated. This is a necessary step before understanding whether or not the techniques described in this section can be employed in CLAM or not, and is left as a future line of research.

As a final note, we remark that event data presents temporal, spatial, and spatio-temporal correlation, which can be utilized for event detection process to enable situational awareness. In this deliverable, we only look into networking protocols to disseminate data, and thereby investigate the effectiveness of transmission scheduling techniques in hierarchical underwater networks employing data aggregation. The use of correlation among sensor data and neighboring nodes and the benefit it offers for event detection and situational awareness will be dealt with in the upcoming deliverables.

11 Discussion and Conclusions

This deliverable presents the initial design of collaborative networking and processing protocols in CLAM. Several cross-layer approaches have been devised for channel access, and routing. These approaches include link-level error control policies. All protocols are at an advanced stage of development, although at this stage of the project it is natural that some aspects need be fine-tuned (e.g., the vulnerability exhibited by the SUN routing protocol at high traffic). Several quantitative results obtained with the nsMiracle+WOSS simulation software have been presented and compared in this deliverable.

An aspect of the workflow so far is worth being stressed: the protocols can be transported almost straightforwardly into the CLAM node using the framework in WP5 (where “almost” means that the physical layer module to be used is not WOSS but the module interfacing nsMiracle to the CLAM modem). This also makes it possible to easily bring any future protocol optimization directly into the nodes. The interaction between WP4 and WP5 was key in this regard.

In light of the above, and in preparation for the integration work and sea trials in WP5, we employed the some of the protocols studied in this deliverable in a network akin to the likely deployment of the final CLAM demonstration. The results are reported in D5.2.

In WP4, we also did theoretical work. An analytical framework to estimate the performance of underwater networks under optimal scheduling and routing has been deployed. The framework has been further developed and integrated with heuristics that allow it to scale well to networks of large size. The new version of the framework has been shown to yield near-optimal results in all cases where the original framework successfully found an optimal solution.

The cooperation with WP3 has been key for performing the simulations of UW-Polling, one of the protocols presented in this deliverable. In fact, UW-Polling simulations can be very complex when using WOSS: as nodes moves, the channel power gains change and must be re-computed. In turn, this requires to run many instances of Bellhop. With a pseudo-analytic approach studied in WP3, it has been possible
to optimize the parameters of an empirical channel model (and in particular the path loss exponent) in such a way that the empirical model yielded channel power gains similar to those provided by Bellhop. In turn, this allows to speed up simulations considerably while providing reliable network performance estimates.

As a final line of work, we have analyzed the effect of scheduling policies on the performance of a hierarchical, aggregation-aware routing protocol for underwater networks. The results suggest that aggregation is a possible future direction to improve the effectiveness of data retrieval, and that the hierarchical structure of the network can be exploited with delay-aware schedules, provided that a shared time reference can be distributed.

References


